Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 15.03.04 «Автоматизация технологических процессов и производств»
Отделение школы (НОЦ) Отделение автоматизации и робототехники

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Создание системы управления электродвигателем на базе микроконтроллера STM32

УДК 004.896:004.31:621.313.13

Студент

Группа	ФИО	Подпись	Дата
8Т4Б	Житников Алексей Дмитриевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Фадеев Александр Сергеевич	к.т.н.		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель ОСГН	Хаперская Алена Васильевна	—		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент ИШХБМТ	Невский Егор Сергеевич	—		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата

Томск – 2018 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ООП

Код результата	Результат обучения (Выпускник должен быть готов)
	Профессиональные компетенции
Р1	Демонстрировать базовые естественнонаучные и математические знания для решения научных и инженерных задач в области анализа, синтеза, проектирования, производства и эксплуатации систем автоматизации технологических процессов и производств. Уметь сочетать теорию, практику и методы для решения инженерных задач, и понимать область их применения.
Р2	Иметь осведомленность о передовом отечественном и зарубежном опыте в области теории, проектирования, производства и эксплуатации систем автоматизации технологических процессов и производств.
Р3	Применять полученные знания для определения, формулирования и решения инженерных задач при разработке, производстве и эксплуатации современных систем автоматизации технологических процессов и производств с использованием передовых научно-технических знаний и достижений мирового уровня, современных инструментальных и программных средств.
Р4	Уметь выбирать и применять соответствующие аналитические методы и методы проектирования систем автоматизации технологических процессов и обосновывать экономическую целесообразность решений.
Р5	Уметь находить необходимую литературу, базы данных и другие источники информации для автоматизации технологических процессов и производств.
Р6	Уметь планировать и проводить эксперимент, интерпретировать данные и их использовать для ведения инновационной инженерной деятельности в области автоматизации технологических процессов и производств.
Р7	Уметь выбирать и использовать подходящее программно-техническое оборудование, оснащение и инструменты для решения задач автоматизации технологических процессов и производств.
	Универсальные компетенции
Р8	Владеть иностранным языком на уровне, позволяющем работать в интернациональной среде с пониманием культурных, языковых и социально-экономических различий.
Р9	Эффективно работать индивидуально, в качестве члена и руководителя группы с ответственностью за риски и работу коллектива при решении инновационных инженерных задач в области автоматизации технологических процессов и производств, демонстрировать при этом готовность следовать профессиональной этике и нормам.
Р10	Иметь широкую эрудицию, в том числе знание и понимание современных общественных и политических проблем, вопросов безопасности и охраны здоровья сотрудников, юридических аспектов, ответственности за инженерную деятельность, влияния инженерных решений на социальный контекст и окружающую среду.
Р11	Понимать необходимость и уметь самостоятельно учиться и повышать квалификацию в течение всего периода профессиональной деятельности.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 15.03.04 «Автоматизация технологических процессов и производств»
Отделение школы (НОЦ) Отделение автоматизации и робототехники

УТВЕРЖДАЮ:
Руководитель ООП

_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

бакалаврской работы

Студенту:

Группа	**ФИО**
8Т4Б	Житникову Алексею Дмитриевичу

Тема работы:

Система управления частотой электродвигателя на базе микроконтроллера STM32	
Утверждена приказом директора (дата, номер)	№2183/с от 28.03.2018 г.

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	– Техническая документация по микроконтроллеру STM32 – Техническая документация по преобразователю частоты на базе микроконтроллера C2000 – Интернет-ресурсы, включающие описания протоколов Modbus и UART – Интернет-ресурсы по проектированию и разработке устройств в программах STM32CubeMX и Keil uVision

Перечень подлежащих исследованию, проектированию и разработке вопросов	Компоненты системы управления двигателем • Оборудование системы управления • Программное обеспечение для создания и отладки алгоритмов • Протоколы обмена данными Реализация системы управления электродвигателем • Реализация UART протокола • Реализация протокола Modbus RTU • Реализация функционала устройства сопряжения
Перечень графического материал	Презентация в формате *.ppt

Консультанты по разделам выпускной квалификационной работы

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Хаперская Алена Васильевна
Социальная ответственность	Невский Егор Сергеевич

Названия разделов, которые должны быть написаны на русском и иностранном языках:

—

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Фадеев Александр Сергеевич	к.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8Т4Б	Житников Алексей Дмитриевич		

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 15.03.04 «Автоматизация технологических процессов и производств»
Уровень образования Бакалавриат
Отделение школы (НОЦ) Отделение автоматизации и робототехники
Период выполнения осенний / весенний семестр 2017/2018 учебного года

Форма представления работы:

бакалаврская работа

(бакалаврская работа, дипломный проект/работа, магистерская диссертация)

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
15.05.2018	Основная часть	75
19.05.2018	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	15
29.05.2018	Социальная ответственность	10

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Фадеев Александр Сергеевич	к.т.н.		

СОГЛАСОВАНО:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8Т4Б	Житникову Алексею Дмитриевичу

Школа	ИШИТР	Отделение	ОАР
Уровень образования	Бакалавриат	Направление/специальность	15.03.04 «Автоматизация технологических процессов и производств»

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	Оклады ТПУ, компоненты системы на сайте на сайте chipdip.ru
2. *Нормы и нормативы расходования ресурсов*	Расчетно-аналитические
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	Общая система налогообложения с учётом льгот для учебных учреждений (в том числе 27,1% отчислений во внебюджетные фонды)
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения*	1.Потенциальные потребители результатов исследования 2.Анализ конкурентных технических решений 3.SWOT – анализ
2. *Планирование и формирование бюджета научных исследований*	1.Основная заработная плата исполнительной темы 2.Дополнительная заработная плата исполнительной темы 3.Отчисления во внебюджетные фонды 4.Расчет материальных затрат 5.Прочие расходы 6.Формирование бюджета затрат НТИ
3. *Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования*	Определение ресурсоэффективности проекта расчётом интегрального показателя
Перечень графического материала *(с точным указанием обязательных чертежей)*:	
1. *Оценка конкурентоспособности технических решений* 2. *Матрица SWOT* 3. *График проведения и бюджет НИ* 4. *Оценка ресурсной, финансовой и экономической эффективности НИ* 5. *Карта сегментирования рынка*	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель	Хаперская Алена Васильевна	—		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8Т4Б	Житников Алексей Дмитриевич		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА

«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8Т4Б	Житникову Алексею Дмитриевичу

Школа	ИШИТР	Отделение	ОАР
Уровень образования	Бакалавриат	Направление/специальность	15.03.04 «Автоматизация технологических процессов и производств»

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Система управления частотой электродвигателя на базе микроконтроллера STM32
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Структура проекта	Рассмотрены основные задачи проекта и его состав. Приведены структурные схемы проекта.
2. Краткое описание протокола Modbus RTU	Рассмотрены основные понятия протокола, а также принципы его работы и возможные улучшения.
3. Помехоустойчивое кодирование в Modbus RTU	Рассмотрено применение циклического избыточного кода, для проверки информации на целостность и идентификации ошибок при передачи данных.
4. Функция разгрузки информационного потока	Рассмотрена функция, реализованная на STM32, созданная для разгрузки ведомых устройств от избыточной информации.
5. Резервирование и самодиагностика	Рассмотрены методы повышения надежности проекта за счет резервирования и самодиагностика.

Дата выдачи задания для раздела по линейному графику	01.03.2018

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент ИШХБМТ	Невский Егор Сергеевич	—		01.03.2018

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8Т4Б	Житников Алексей Дмитриевич		01.03.2018

Реферат

Выпускная квалификационная работа выполнена на 95 с., содержит 30 рисунков, 21 таблицу, 6 приложений.

Ключевые слова: микроконтроллер STM32, электродвигатель, система управления, протокол Modbus, протокол UART, устройство сопряжения.

Объектом исследования является система управления электродвигателем на базе микроконтроллера STM32.

Цель работы — разработка системы управления частотой электродвигателя на базе микроконтроллера STM32.

В работе рассмотрены компоненты системы управления. Для сопрягающего устройства проведен сравнительный анализ микроконтроллеров. Рассмотрено программное обеспечение для программирования микроконтроллера STM32, а также протоколы обмена данными UART и Modbus. Показаны этапы создания сопрягающего устройства между диспетчерской станцией и преобразователем частоты. Проведено тестирование и отладка сопрягающего устройства.

Результатом выполненной работы является созданное сопрягающее устройство, осуществляющее обмен данными между диспетчерской станцией и преобразователем частоты.

Актуальность работы обусловлена растущей популярностью микропроцессорной техники, в частности микроконтроллеров.

Достоинством работы является снижение стоимости системы управления за счет внедрения микроконтроллера в технологический процесс управления электродвигателем вместо коммутирующих устройств и промышленных контроллеров.

Оглавление

Введение

В настоящее время происходит интенсивное развитие микропроцессорной техники, в частности микроконтроллеров (МК) [1]. Все больше они выступают в качестве устройств управления для домашних систем автоматизации, которые могут включать в себя контроль освещения, сигнализацию, дистанционное управление электроприборами и т.д.

Использование таких систем имеет ряд значительных достоинств, таких как: дешевизна самих микроконтроллеров, малое энергопотребление, быстрота разработки новых систем и модернизации старых, возможность расширения систем. Однако, микроконтроллеры имеют ряд недостатков:

- Низкая надежность систем, построенных на таких микроконтроллерах, в сравнении с системами, построенными на базе промышленных контроллеров, обусловленная влиянием окружающей среды [1]. Большое влияние на стабильность работы оказывают температура, электромагнитные помехи, радиация. Особенно чувствительны контроллеры к электромагнитным воздействиям, которые вызывают зависания и самопроизвольные перезагрузки.
- Требование профессиональных компетенций для разработки систем. Для написания качественного кода требуется большой объем знаний в области программирования, в отличие от написания программы на персональном компьютере (ПК), микроконтроллер гораздо чувствительнее к ошибкам программиста [1]. Ограниченный объем памяти, требования к быстродействию «по тактам» требуют высокой квалификации разработчика, также нужно уметь предвидеть различные ситуации, которые возникнут непосредственно в ходе протекания процесса.

Для задач управления опасным для жизни людей технологическим процессом, например, для задачи изменения и контроля положения управляющих стержней ядерного реактора, устойчивость систем к отказам и сбоям является очень важным фактором, именно поэтому в производстве используются промышленные контроллеры, обладающие высокой надежностью.

Однако, принимая во внимание стоимость промышленных контроллеров, и технологический процесс, в регулировании которого они участвуют, использование таких контроллеров не всегда целесообразно. В свою очередь, микроконтроллер, при грамотном подходе к проектированию, может функционально заменить промышленный контроллер и обеспечить необходимый уровень надежности.

Одной из частых задач на производстве является управление насосами, вентиляторами, компрессорами, станками и другими механизмами, большинство из которых содержит электропривод с электродвигателем постоянного или переменного тока, для которых важно поддерживать скорость вращения вала двигателя, либо определенный технологический параметр [2].

Основным элементом современного электропривода является система управления электродвигателем: частотный преобразователь или сервопривод. Преобразователь частоты (ПЧ) позволяет управлять моментом и скоростью вращения электродвигателя и исполнительного механизма. Сервопривод позволяет точно управлять угловым положением, скоростью и ускорением исполнительного механизма [3].

Связь частотного преобразователя и ПК может быть выполнена по промышленному протоколу Modbus RTU, который является наиболее широко используемым протоколом во всей промышленной автоматизации по ряду причин:

- Modbus — протокол с открытым исходным кодом. Это означает, что он может быть включен в широкий диапазон типов устройств от любого поставщика оборудования;
- Modbus использует простую структуру сообщений, что делает ее менее сложной для развертывания и отладки. Это является конкурентным преимуществом в сравнении с другими протоколами, которые могут потребовать месяцев для изучения и развертывания;
- Modbus используется с двумя типами последовательных соединений: RS-232 и RS-485.

В данной работе рассматривается применение микроконтроллера STM32, как связывающего устройства между преобразователем частоты и персональным компьютером, осуществляющее передачу данных между ними посредством протокола Modbus RTU.

Цель работы:

- разработка системы управления частотой электродвигателя на базе микроконтроллера STM32.

Задачи работы:

- изучить структуру системы управления частотой электродвигателя;
- изучить среду разработки Keil MDK-ARM и программное обеспечение (ПО) для конфигурирования STM32 — STM32CubeMX;
- реализовать протокол передачи данных Modbus RTU;
- разработать программу, осуществляющую связь между частотным преобразователем и ПК посредством протокола Modbus RTU;
- протестировать управление частотой вращения электродвигателя через ПК.

1. Компоненты системы управления двигателем

В данном обзоре рассмотрены устройства, необходимые для создания системы управления двигателем. Приведено программное обеспечение, необходимое для реализации и отладки системы, а также рассмотрены протоколы приема/передачи данных, на которых базируется система.

1.1. Оборудование системы управления

Структурная схема системы управления частотой двигателя показана на рисунке 1.

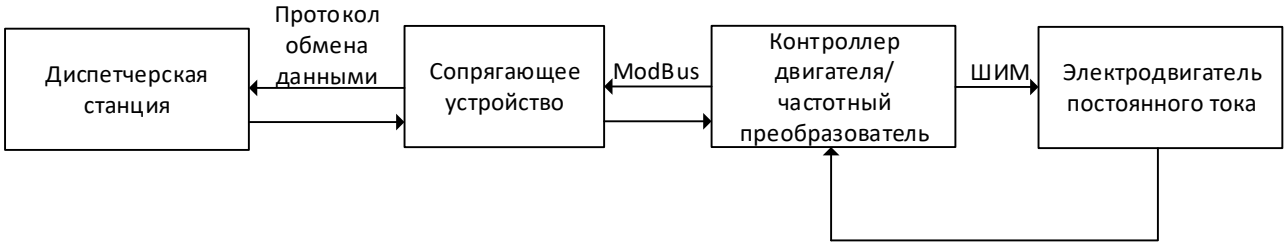


Рисунок 1– Структурная схема системы управления электродвигателем

С диспетчерской станции на сопрягающее устройство поступают команды управления преобразователем частоты. Команды с диспетчерской станции могут приходить по различным протоколам, функция сопрягающего устройства состоит в преобразовании протокола данных с диспетчерской станции в протокол Modbus. Далее данные поступают на частотный преобразователь, который преобразует входные данные в токовый сигнал управления электродвигателем.

В реализованной системе в качестве диспетчерской станции выступает персональный компьютер. Для упрощения реализации системы на первых этапах, команды с ПК на сопрягающее устройство поступают сразу по протоколу Modbus, тогда в функционал устройства входит проверка данных на целостность и информационная разгрузка подключаемых устройств.

1.1.1. Сопрягающее устройство

Учитывая функции сопрягающего устройства, было принято решение использовать для его реализации микроконтроллер. Промышленные контроллеры в качестве сопрягающего устройства не рассматривались. Их надежность и технические характеристики превосходят микроконтроллеры, но и цена тоже име-

ет существенные отличия. Также, использование для таких целей микроконтроллеров представляет широкий интерес для разработчиков встраиваемых систем.

Для выбора сопрягающего устройства был проведен сравнительный анализ между двумя микроконтроллерами ATmega328P на базе аппаратной платформы Arduino Nano (рисунок 2) и STM32F103C8T6 (рисунок 3).

Рисунок 2 – Arduino Nano

Рисунок 3 – STM32F103C8T6

1.1.1.1. Технические характеристики микроконтроллеров

Технические характеристики микроконтроллеров сведены в таблицу 1 [5].

Таблица 1– Технические характеристики STM32F103C8T6 и Arduino Nano

Характеристики	STM32F103C8T6	Arduino Nano
Частота микроконтроллера, МГц	72	16
Память программ, кБайт	64	32
Питание, В	3.3	5
ОЗУ, кБайт	20	2

Характеристики	STM32F103C8T6	Arduino Nano
USB 2.0	+	-
DMA	+	-
CAN	+	-
RTC	+	-
UART	3	1
Прошивка через USB	-	+

Анализ технических характеристик показал, что STM32F103 превосходит Arduino Nano практически по всем показателям. Особенно большие различия наблюдаются в быстродействии, характеризуемом частотой микроконтроллера и в возможностях подключения периферийных устройств, характеризуемых количеством каналов ввода-вывода:

- тактовая частота STM32 превышает частоту оппонента в 4,5 раза, то есть система, созданная на базе STM32, заранее имеет более высокие показатели быстродействия;
- UART (Universal asynchronous receiver/transmitter) — универсальный асинхронный приемопередатчик, физический протокол передачи данных [6]. Наличие трех UART каналов на плате STM32 позволяет отлаживать созданный проект, не используя дополнительных плат. В случае с Arduino Nano понадобилось бы подключать модули расширения, что привело бы к повышению стоимости комплектующих проекта и дополнительным затратам энергоресурсов.

Также STM32 имеет модуль DMA (Direct Memory Access — прямой доступ к памяти — режим обмена данными между устройствами компьютера или же между устройством и основной памятью, в котором центральный процессор (ЦП) не участвует. Так как данные не пересылаются в ЦП и обратно, скорость передачи увеличивается) [7]. Данный модуль будет применен при дальнейшей разработке системы и увеличении ее быстродействия.

Отсутствие возможности программировать STM32 через USB-порт компенсируется использованием программатора JTAG, который имеет высокоско-

ростной интерфейс для подключения к ПК, а также обеспечивает питание самой плате.

1.1.1.2. Достоинства и недостатки микроконтроллеров Arduino и STM32

Одно из главных достоинств Arduino – собственная «экосистема». Огромное количество датчиков и модулей расширения позволяет создавать проекты высокой сложности и для широкого спектра нужд. Информационная база по Arduino включает в себя множество русскоязычных ресурсов с подробно описанными проектами и детальными инструкциями. Большое количество пользователей Arduino подразумевает и широкий программный функционал: ежедневно создаются десятки библиотек для различного применения аппаратной платформы [8].

Однако, данные преимущества Arduino влекут за собой и недостатки:

- большое количество неотлаженных библиотек, неэффективно использующих память микроконтроллера и работающих не всегда правильно;
- в сравнении с STM32, отсутствие хорошего отладчика, из-за чего поиск и устранение ошибок занимает большое количество времени;
- отсутствие избыточной профессиональной информации. Подавляющее количество информации представляется на форумах, самими пользователями [8].

Преимущество Arduino перед STM32 очевидно — данная аппаратная платформа подходит для начала изучения мира микроконтроллеров и создания небольших проектов в сфере домашней автоматизации и робототехники. Однако, для создания проектов, участвующих в сфере промышленной автоматизации, большее внимание уделяется не наличию написанных библиотек и русскоязычной литературы, а повышенной надежности и более высокой производительности микроконтроллеров. Технические преимущества STM32 не оставляют сомнений, также микроконтроллер обладает более высокой стабильностью, связанной с тем, что ядро и периферия микроконтроллера работают от напря-

жения 1.8В, при напряжении питания 3.3В, когда периферия микроконтроллера ATmega328 работает от напряжения 2.7В, при напряжении питания 5В.

Также, большое преимущество STM32 — разнообразие инструментов разработки и отладки. Это касается как аппаратных, так и программных средств [9].

Исходя из всех вышеперечисленных аргументов, было сделано заключение, что для данного проекта подходит микроконтроллер STM32F103C8T6.

1.1.2. Преобразователь частоты вращения электродвигателя

В качестве преобразователя частоты используется устройство для управления двигателями постоянного тока, использующего микроконтроллер TMS320F2803x, относящийся к семейству микроконтроллеров C2000. В данном устройстве встроены алгоритмы динамического управления двигателями, использующие широтно-импульсную модуляцию. Высокая производительность микроконтроллера позволяет адаптировать алгоритмы управления к изменениям в системе в режиме реального времени и одновременно с этим отслеживать параметры электродвигателя [10].

1.1.3. Организация каналов связи между устройствами

Для создания канала связи между персональным компьютером и STM32 используется плата FTDI232, выполняющая функцию переходника между интерфейсами UART и RS485, позволяющая, используя минимум внешних компонентов (разъем и пассивные компоненты), организовать последовательный обмен данными между внешним устройством на микроконтроллере и компьютером через шину USB (рисунок 4).

Рисунок 4 – плата FTDI232

Информация с сопрягающего устройства передается на преобразователь частоты по протоколу UART и не требует дополнительных схемотехнических решений.

1.2. Программное обеспечение для создания и отладки алгоритмов работы на STM32

Программирование микроконтроллера STM32 состоит из трех этапов:

1. Конфигурирование периферии микроконтроллера. Выполняется в программе STM32CubeMX.
2. Написание кода основной программы. Выполняется в среде разработке Keil uVision.
3. Отладка работы устройства. Выполняется в программах Terminal v1.9b, Viewer, а также с помощью встроенного отладчика среды разработки Keil uVision.

Далее приведено описание возможностей и функционала всех вышеперечисленных программ.

1.2.1. Графический генератор кода STM32CubeMX

Генераторы программного кода предназначены для того, чтобы создавать отдельные файлы исходного кода или всего проекта на основе заданных настроек. Чаще всего генераторы используют популярные библиотеки, что позволяет унифицировать процесс генерации исходных текстов. В зависимости от количества настраиваемых параметров они могут за счет реализации части

функционального обеспечения значительно минимизировать время, затрачиваемое на создание основы для дальнейшей разработки. Практика использования подобных генераторов для разработки прикладного ПО во многих отраслях стала стандартной, а знание соответствующих инструментов является обязательным. В то же время, в сфере разработки встраиваемых систем подобные генераторы применяются благодаря преимуществам, которые они предоставляют, однако все еще находятся на этапе развития и получения признания среди разработчиков.

STM32CubeMX - это графический инструмент генерации низкоуровневого кода для микроконтроллеров STM32 [11].

STM32CubeMx — программный продукт, позволяющий при помощи понятного графического интерфейса произвести настройку любой имеющейся на борту микроконтроллера периферии. STM32CubeMx является единым инструментом для настройки и конфигурирования большого количества микроконтроллеров и, сопутствующим им, библиотекам и документациям [12].

Визуальный редактор настроек STM32CubeMX является составной частью инфраструктуры STM32Cube, которая также включает специализированные библиотеки для работы с различными линейками микроконтроллеров STM32.

Основными функциями, которые предоставляет визуальный редактор конфигурации, являются:

- настройка использования выводов периферийными устройствами в составе микроконтроллера (рисунок 5);

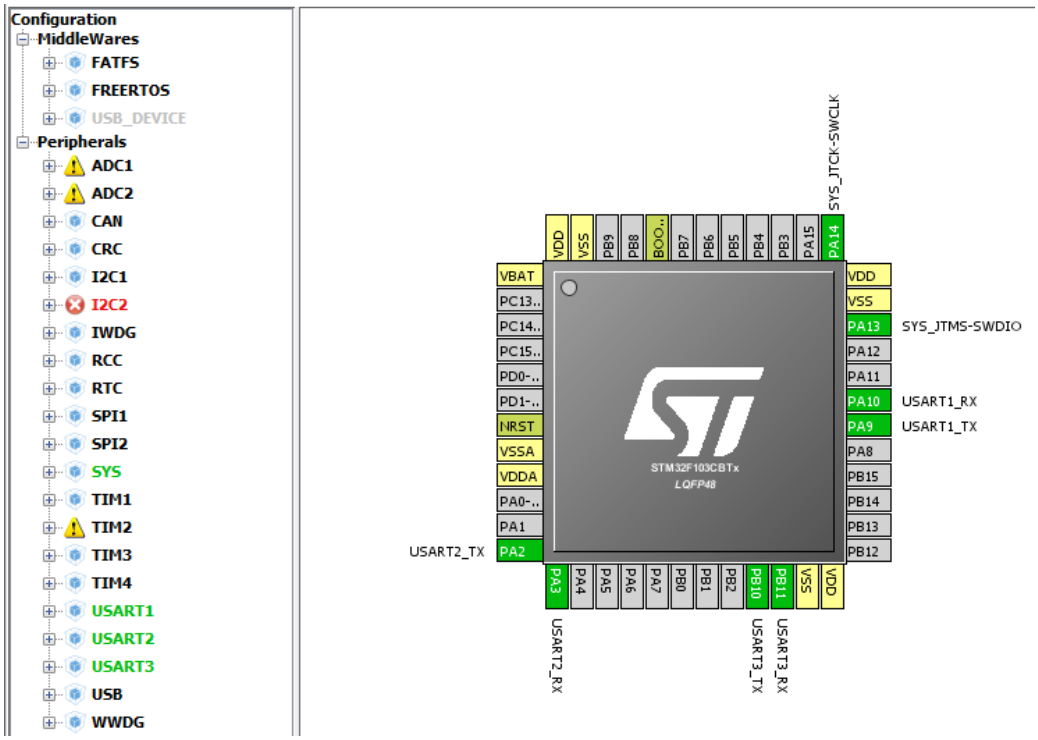


Рисунок 5 – Окно настройки выводов микроконтроллера

- настройка тактирования (рисунок 6);

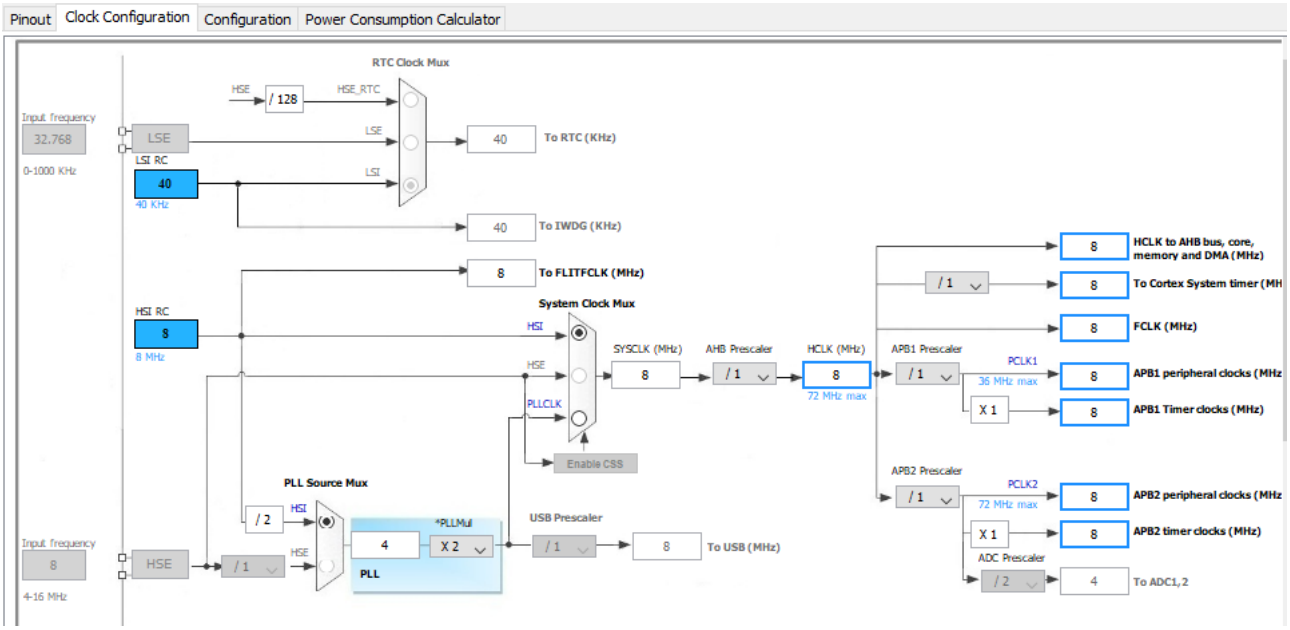


Рисунок 6 – Окно настройки тактирования

- настройка периферийных устройств;
- расчет потребления энергии в соответствии с планируемыми режимами работы.

Интерфейс самой программы, несмотря на наличие большого количества настроек, прост и понятен. Он не перегружен ненужными функциями, а визу-

альное представление различных систем МК позволяет полностью сложить картину о конечном результате выполненных настроек.

Пакет программного обеспечения, который предлагается для использования, включает в себя следующие основные компоненты:

- библиотеку CMSIS;
- библиотеку для работы с периферийными устройствами (Hardware Abstraction Layer – HAL);
- библиотеки, упрощающие использование возможностей популярных плат (Board Support Package – BSP);
- библиотеки более высокого прикладного уровня, позволяющие реализовать специализированные возможности (система реального времени FreeRTOS, поддержка файловой системы FAT, работа со стеком протоколов TCP/IP благодаря библиотеке lwIP, использование возможностей USB и др.). Не все библиотеки данного уровня могут быть доступны для подключения при использовании конкретного микроконтроллера.

К недостаткам генераторов кода можно отнести ограниченное количество настроек, которое не позволяет учесть все особенности проекта. Также при разработке проекта для динамической системы, которая часто изменяет параметры собственной работы в зависимости от внутренних и внешних условий, генераторами можно описать лишь ограниченное количество состояний данной системы. Необходимость учета большого количества критериев влечет за собой внесение большого количества изменений в программную реализацию [13].

После настройки периферии и генерации кода можно приступать непосредственно к разработке и отладке основных алгоритмов работы устройства.

1.2.2. Среда разработки — Keil uVision.

Для программирования устройства была выбрана программная среда Keil uVision.

Keil uVision — среда разработки, представляющая собой набор утилит для выполнения полного комплекса мероприятий по написанию программного обеспечения для микроконтроллеров (рисунок 7).

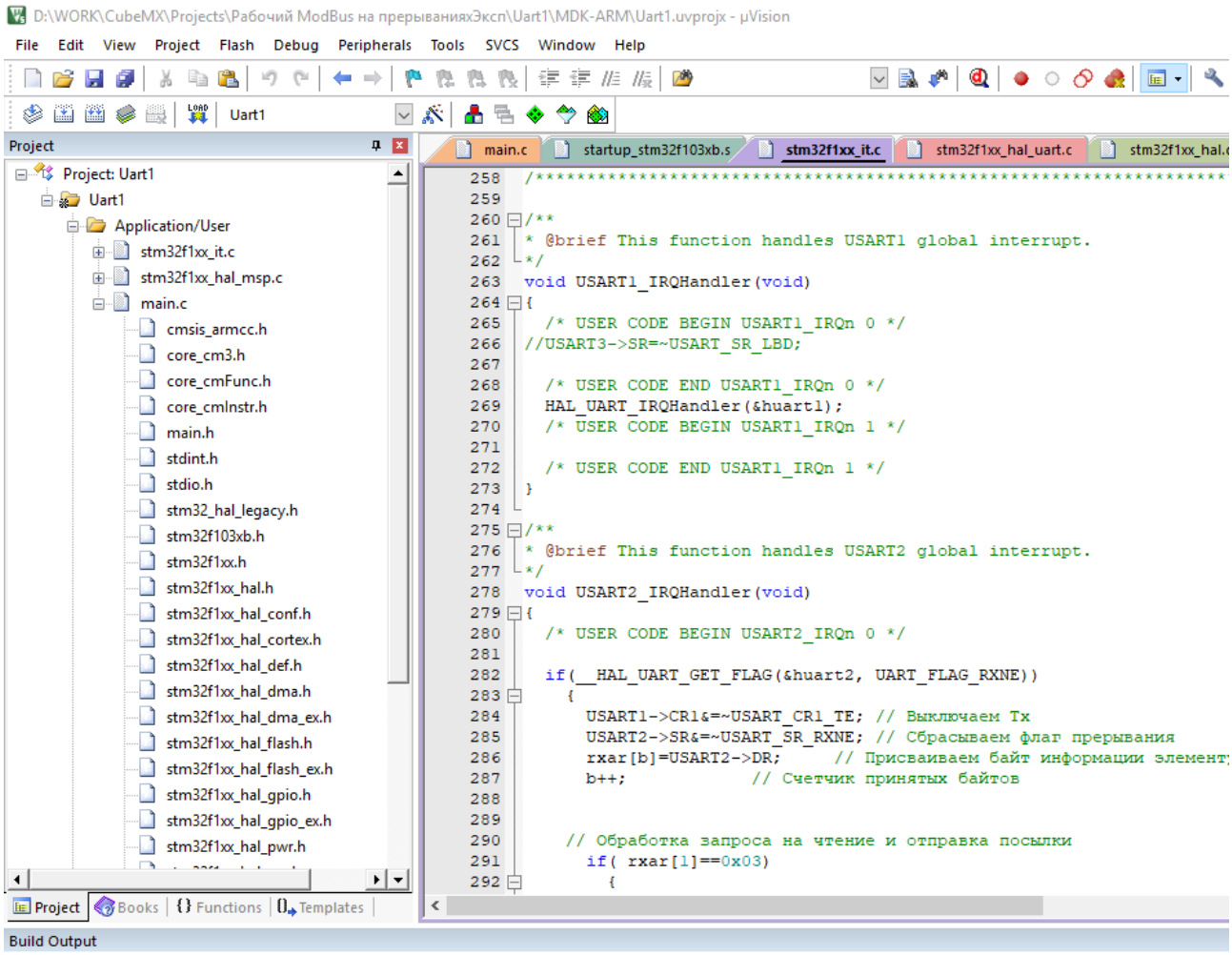


Рисунок 7 – Среда разработки Keil uVision

Среда программирования разработана компанией Keil, которая была основана в Мюнхене в 1982 году братьями Гюнтером и Рейнхардом. В октябре 2005 года Keil вошла в состав американской корпорации ARM. На сегодняшний день она представляет широкий спектр различных средств для разработки программ, включающих Си-компиляторы, макроассемблеры, отладчики, симуляторы, линкеры, IDE-приложения и оценочные платы для различных семейств микроконтроллеров.

Keil uVision позволяет работать с проектами любой степени сложности, начиная с введения и правки исходных текстов и заканчивая внутрисхемной отладкой кода и программированием ПЗУ микроконтроллера. От разработчика

скрыта большая часть второстепенных функций, что сильно разгружает интерфейс и делает управление интуитивно понятным. Однако при возрастании сложности реализуемых задач, всегда можно задействовать весь потенциал модулей, функционирующих под управлением единой оболочки.

Среди основных программных средств Keil uVision можно отметить:

1. Базу данных микроконтроллеров, содержащую подробную информацию обо всех поддерживаемых устройствах. Здесь хранятся их конфигурационные данные и ссылки на источники информации с дополнительными техническими описаниями. При добавлении нового устройства в проект все его уникальные опции устанавливаются автоматически.

2. Менеджер проектов, служащий для объединения отдельных текстов программных модулей и файлов в группы, обрабатываемые по единым правилам. Подобная группировка позволяет намного лучше ориентироваться среди множества файлов.

3. Встроенный редактор, облегчающий работу с исходным текстом за счет использования многооконного интерфейса, выделения синтаксических элементов шрифтом и цветом. Существует опция настройки в соответствии со вкусами разработчика. Редактирование остается доступным и во время отладки программы, что позволяет сразу исправлять ошибки или отмечать проблемные участки кода.

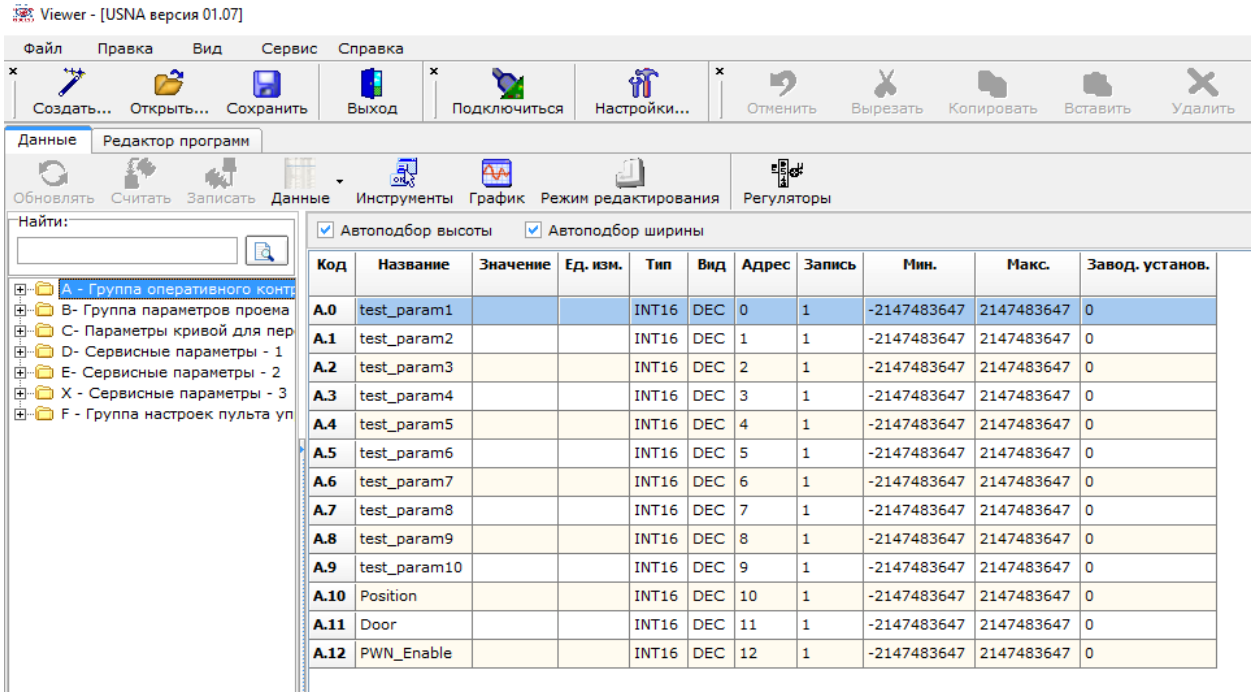
4. Средства автоматической компиляции, ассемблирования и компоновки проекта, которые предназначены для создания исполняемого (загрузочного) модуля программы. При этом между файлами автоматически генерируются новые ассемблерные и компиляторные связи, которые в дальнейшем позволяют обрабатывать только те файлы, в которых произошли изменения или файлы, находящиеся в зависимости от изменённых. Функция глобальной оптимизации проекта позволяет достичь наилучшего использования регистров микроконтроллера путем неоднократной компиляции исходного кода. Компиляторы uVision работают с текстами, написанными на Си или ассемблере для контрол-

леров семейств ARM, MSC51, C166 и многих других. Кроме того, возможно использование компиляторов других производителей.

5. Отладчик-симулятор, отлаживающий работу скомпилированной программы на виртуальной модели микропроцессора. Довольно достоверно моделируется работа ядра контроллера и его периферийного оборудования: портов ввода-вывода, таймеров, контроллеров прерываний. Для облегчения комплексной отладки разрабатываемого программного обеспечения возможно подключение программных моделей нестандартного оборудования [14].

1.2.3. Программа обмена данными по протоколу Modbus — Viewer

Программа Viewer (рисунок 8), созданная компанией «ЭлиСи» осуществляет соединение персонального компьютера с устройством, обменивается данными с подключённым устройством по протоколу Modbus, использует функции записи данных в устройство и чтения данных из устройства. Функция чтения данных позволяет отслеживать параметры технологического процесса в реальном времени. С помощью данной программы производится отладка работы протокола Modbus на микроконтроллере, а также осуществляется управление электродвигателем.



Код	Название	Значение	Ед. изм.	Тип	Вид	Адрес	Запись	Мин.	Макс.	Завод. установ.
A.0	test_param1			INT16	DEC	0	1	-2147483647	2147483647	0
A.1	test_param2			INT16	DEC	1	1	-2147483647	2147483647	0
A.2	test_param3			INT16	DEC	2	1	-2147483647	2147483647	0
A.3	test_param4			INT16	DEC	3	1	-2147483647	2147483647	0
A.4	test_param5			INT16	DEC	4	1	-2147483647	2147483647	0
A.5	test_param6			INT16	DEC	5	1	-2147483647	2147483647	0
A.6	test_param7			INT16	DEC	6	1	-2147483647	2147483647	0
A.7	test_param8			INT16	DEC	7	1	-2147483647	2147483647	0
A.8	test_param9			INT16	DEC	8	1	-2147483647	2147483647	0
A.9	test_param10			INT16	DEC	9	1	-2147483647	2147483647	0
A.10	Position			INT16	DEC	10	1	-2147483647	2147483647	0
A.11	Door			INT16	DEC	11	1	-2147483647	2147483647	0
A.12	PWN_Enable			INT16	DEC	12	1	-2147483647	2147483647	0

Рисунок 8 – Основное окно программы Viewer

1.2.4. Terminal v1.9b

Программа Terminal v1.9b (рисунок 9) позволяет настраивать соединение с устройством по UART протоколу, отслеживать принятые через USB-порт данные, а также отправлять данные на подключенное устройство, просматривать полученные или отправленные сообщения с точностью до битов. В данном проекте программа используется для отладки протокола физического уровня — UART.

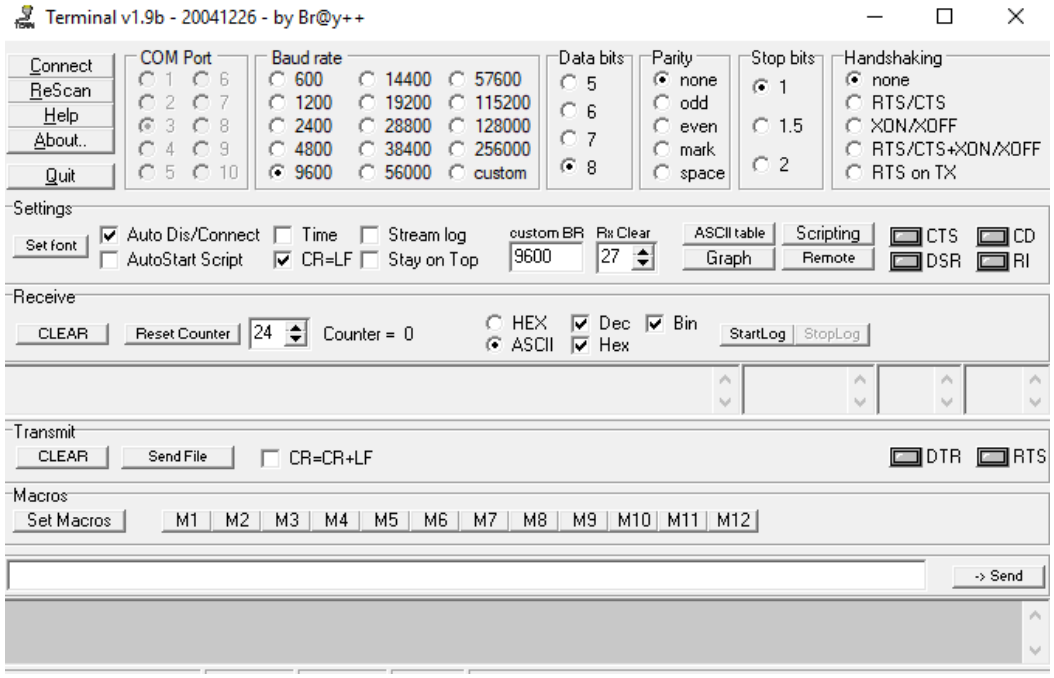


Рисунок 9 – Программа Terminal v1.9b

1.3. Протоколы обмена данными

Процесс обмена данными системы основан на протоколах Modbus RTU и UART.

Под протоколом UART подразумевается физический уровень передачи данных протокола Modbus.

Далее представлено описание структуры протоколов.

1.3.1. Протокол физического уровня передачи данных — UART

UART — интерфейс для связи цифровых устройств, предназначенный для передачи данных в последовательной форме. Является очень распространённым и имеет аппаратную реализацию во многих микроконтроллерах. Микроконтроллер STM32F103C8T6 содержит три USART.

USART (Universal Synchronous-Asynchronous Receiver/Transmitter) - универсальный синхронно-асинхронный приёмопередатчик - аналогичный UART интерфейс, но дополнительно к возможностям UART, поддерживает режим синхронной передачи данных - с использованием дополнительной линии тактового сигнала. Впрочем, синхронная передача используется гораздо реже асинхронной, и в нашем случае, посредством STM32CubeMX, USART будет использовать режим асинхронной передачи данных.

UART может использоваться как для взаимодействия компонентов внутри одного устройства, так и для подключения устройств между собой. Распространёнными стандартами физического уровня для UART, которые подходят для подключения внешних устройств является RS232/485/422. Этим стандартам соответствует последовательный порт (COM-порт) компьютера. Так что, микроконтроллер с помощью схемы преобразования уровней может обмениваться информацией с COM-портом компьютера.

1.3.1.1. Формат передачи данных по UART протоколу

В отсутствии передачи на выходе UART присутствует уровень логической единицы. Данные передаются в виде посылок (фреймов), каждая из которых состоит из стартового бита, битов данных и одного или нескольких стопбитов (рисунок 10). Длительность всех битов одинакова, связана со скоростью передачи соотношением T=1/S. Существует ряд стандартных скоростей передачи: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 бод. Если внутри одного устройства связь можно осуществлять на произвольной скорости, то для связи с внешними устройствами следует придерживаться стандартных величин.

Рисунок 10 – Формат посылки UART

Посылка начинается со стартового бита, он всегда имеет значение логического нуля. После стартового бита передаются биты данных. Количество битов данных может составлять 5-9 в зависимости от настроек UART. Обычно передаётся 8 бит данных или 9 бит (8 бит собственно данных и один бит чётности). Завершается посылка стоп-битами, их значение - всегда логическая единица, количество обычно составляет 1, 1.5 или 2. Под количеством стоп-битов понимается длительность соответствующего им единичного импульса по отношению к длительности битов данных и старт-бита. Этим объясняется возможность выражать количество битов дробным числом. Сразу же после стоп-битов может начинаться передача следующей посылки или может быть пауза произвольной длительности, во время которой на выходе также формируется уровень логической единицы.

Так как во время передачи стоп-бита и пока линия свободна, на выходе присутствует единичное значение, а старт-бит имеет значение логического нуля, старт-бит позволяет выявить момент начала передачи данных, разделить две последовательные посылки и осуществить синхронизацию передатчика и приёмника.

Если передатчик и приёмник работают на одной скорости, настроены на работу с одинаковым количеством битов данных, стоп битов, одинаково сконфигурированы в отношении бита чётности, то для обмена данными не требуется передавать отдельно тактовый сигнал - он может быть восстановлен приёмником самостоятельно [15].

1.3.2. Протокол Modbus RTU

Modbus – открытый протокол обмена данными в малых локальных сетях. Как правило, используется для передачи данных через интерфейсы RS-232, RS-485, RS-422, в сетях TCP/IP, UDP. Благодаря своей простоте и универсальности Modbus получил широкое распространение и стал де факто стандартом в малых распределенных вычислительных системах. Практически все современные контроллеры поддерживают работу в сетях Modbus.

В сети Modbus контроллеры, как правило, соединены по топологии «Общая шина». Взаимодействие контроллеров происходит в соответствии с моделью master-slave (ведущий-ведомый) (рисунок 11).

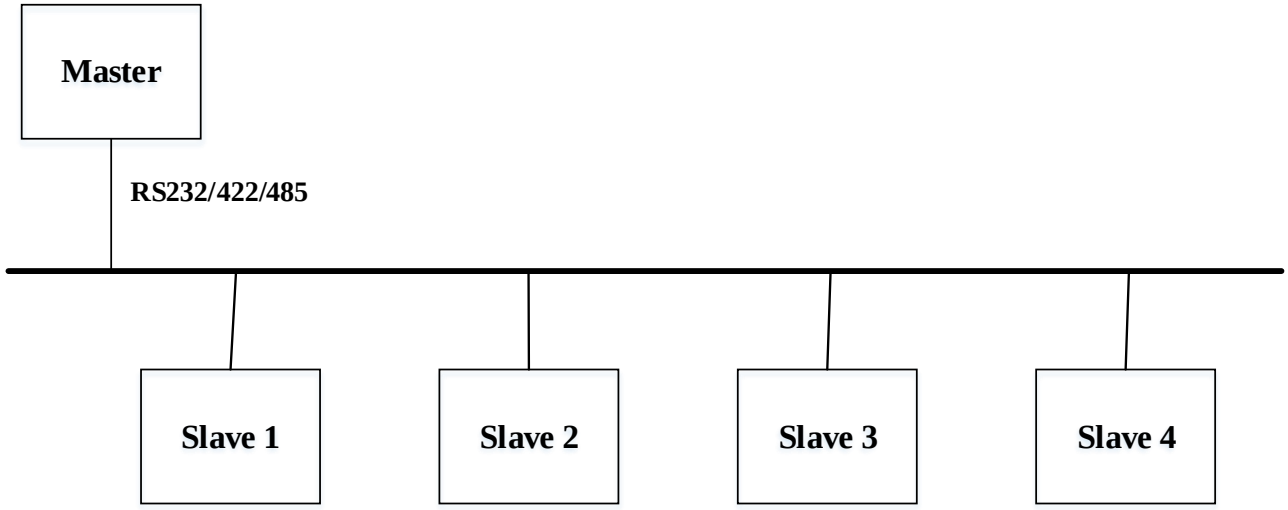


Рисунок 11 – Схема топологии «Общая шина»

В сети есть главное устройство - ведущее. А также несколько подчиненных устройств – ведомые. Обмен может быть инициирован только ведущим устройством.

Транзакция (последовательность операций при обмене данными) состоит из запроса и ответа.

Ведущее устройство может адресовать запрос любому ведомому контроллеру или инициировать широковещательное сообщение, для всех ведомых устройств одновременно.

Ведомое устройство, определив свой адрес в запросе, формирует ответ.

В запросе от ведущего устройства обязательно содержится код функции, т.е. что надо делать. Также в зависимости от функции в запросе могут содержаться данные.

Существуют 3 варианта протокола ModBus:

- ModBus ASCII – текстовый протокол. В нем используются только ASCII символы. Каждый байт передается как два шестнадцатеричных символа;
- ModBus RTU – числовой протокол. Данные передаются в двоичном виде. Байт, передаваемый по сети это число протокола;
- ModBus TCP – протокол для передачи данных в TCP/IP сетях.

1.3.2.1. Формат передачи данных по Modbus RTU протоколу

Ведущее устройство инициирует транзакцию – обмен данными. Транзакция может быть индивидуальной (запрос-ответ) или широковещательной (адресуются одновременно все ведомые устройства). Транзакция состоит из одного кадра запроса и одного кадра ответа. При широковещательной транзакции используется только кадр запроса.

Данные кадра передаются сплошным потоком. Пауза между передачей данных не должна превышать время передачи 1,5 символа. Признаком нового кадра является отсутствие обмена в сети (молчание) в течение времени необходимого на передачу 3,5 символов. Если в течение этого времени линия сети находилась в неактивном состоянии, то ведомые устройства воспринимают первое принятое данное как начало кадра.

В общем случае кадр запроса имеет следующий формат (рисунок 12).

Адрес 8 бит	**Функция 8 бит**	**Данные N*8 бит**	**Контрольная сумма 16 бит**

Рисунок 12 – Кадр запроса ведущего устройства

Кадр начинается с поля адреса, которое состоит из 8 разрядов. Содержит адрес ведомого устройства, которому предназначено сообщение от ведущего. Каждое ведомое устройство должно иметь уникальный адрес от 1 до 247. И

только адресуемое ведомое устройство должно ответить на запрос ведущего. В ответе сообщается ведущему устройству, с каким из ведомых установлена связь.

Адрес 0 используется в широковещательном режиме. Все ведомые устройства выполняют заданную в запросе функцию, но подтверждение не посылают.

Поле функции сообщает адресуемому ведомому устройству, какую операцию выполнить.

Старший бит байта функция устанавливается в 1 в ответе ведомого устройства, чтобы сообщить ведущему, что операция была выполнена с ошибкой. При успешной операции старший бит равен 0.

1.3.2.2. Функции протокола Modbus RTU

Выделяются 3 категории кодов функций:

- стандартные команды. Коды, определенные стандартом на протокол ModBus;
- пользовательские команды. Для кодов 65…72 и 100…110 пользователь может сам назначить произвольную функцию;
- зарезервированные команды. Это коды, которые не были изначально определены стандартом, но уже используются в устройствах разных производителей.

В созданном проекте используются две функции:

- Чтение значений одного или нескольких регистров хранения (READ HOLDING REGISTERS). Код функции – 03;
- Последовательная запись нескольких регистров хранения (FORCE MULTIPLE REGISTERS). Код функции – 10.

Запрос ведущего устройства на чтение данных имеет формат, представленный в таблице 2.

Таблица 2 – Формат запроса ведущего устройства на чтение данных

ID ведомого устройства	Код функции	Начальный адрес регистров		Количество регистров		Контрольная сумма CRC-16	
0x01	0x03	0x00	0x02	0x00	0x01	0xC5	0xCD

Если ведомое устройство успешно приняло запрос, о чем свидетельствует совпадение полученного CRC от ведущего устройства и рассчитанного ведомым, то формат ответа ведомого устройства представлен в таблице 3.

Таблица 3 – Формат ответа ведомого устройства

ID ведомого устройства	Код функции	Количество прочитанных байтов		Значение регистра		Контрольная сумма CRC-16	
0x01	0x03	0x00	0x04	0x12	0xA5	0xA7	0x70

Функция с кодом 10 используется для записи данных в несколько, последовательно расположенных регистров. В запросе передается адрес первого регистра, количество регистров и значения для них. В ответе возвращается начальный адрес и количество измененных регистров.

Формат запроса функции записи регистров хранения представлен в таблице 4.

Таблица 4 – Формат запроса ведомого устройства на запись данных

ID ведомого устройства	Код функции	Адрес начального регистра		Количество регистров		Количество байт	Данные в первый регистр		Данные во второй регистр		CRC-16	
0x01	0x10	0x00	0x02	0x00	0x02	0x04	0x00	0x01	0x00	0x00	0x8D	0x32

При успешной записи данных в регистры, ведомое устройство отправит ответ, представленный в таблице 5.

Таблица 5 – Формат ответа ведомого устройства

ID ведомого устройства	Код функции	Адрес начального регистра		Количество записанных регистров		CRC-16	
0x01	0x10	0x00	0x02	0x00	0x02	0xE0	0x3B

1.3.2.3. Помехоустойчивое кодирование в протоколе Modbus RTU

В процессе передачи и хранения данных неизбежно возникают ошибки. Обнаружение ошибок и их исправление является важной задачей для создания любого канала связи.

Для проверки целостности поступающих данных, протокол Modbus использует CRC.

CRC (Cyclic Redundancy Code - циклический избыточный код) — алгоритм расчёта контрольной суммы для передаваемого сообщения, основанный на полиномиальной арифметике.

Основная идея алгоритма CRC состоит в представлении сообщения в виде огромного двоичного числа, делении его на другое фиксированное двоичное число и использовании остатка от этого деления в качестве контрольной суммы. Получив сообщение, приёмник должен выполнить аналогичное действие и сравнить полученный результат с принятой контрольной суммой. Сообщение считается достоверным, когда выполняется это равенство.

Алгоритм CRC базируется на полиномиальной арифметике, а это означает, что сообщение, делитель и остаток могут быть представлены в виде полиномов с двоичными коэффициентами или в виде строки битов, каждый из которых является коэффициентом полинома. Чтобы выполнить вычисление CRC, необходимо выбрать делитель - полином. Важной характеристикой, определяющей дальнейшие расчёты, является степень полинома. Обычно выбирается степень 16 или 32, так как они являются кратными разрядности регистров современных процессоров, что значительно упрощает реализацию алгоритмов CRC.

Для ускорения расчёта CRC используется табличный алгоритм. В таблицу внесены все возможные 256 значений расчета полинома. Тогда, вместо алгоритма расчета, используется алгоритм поиска значений CRC по готовой таблице, индексами которой являются исходные данные [16].

Код для расчета CRC-16 табличным методом представлен приложении А.

1.4. Выводы по главе 1

Была рассмотрена система управления электродвигателем постоянного тока, а также оборудование, необходимое для создания и функционирования системы. В ходе сравнительного анализа, в качестве сопрягающего устройства был выбран микроконтроллер STM32F103C8T6. Рассмотрено программное обеспечение для отладки работы микроконтроллера и протоколов связи в системе управления. Приведено описание протоколов Modbus и UART.

2. Реализация системы управления электродвигателем

Для постепенного изучения функционала программного обеспечения и периферии микроконтроллера создание системы управления было разделено на три этапа:

1. Реализация UART протокола и отладка его функционирования.
2. Реализация протокола Modbus RTU. Для отладки протокола микроконтроллер выступает ведомым устройством, а персональный компьютер ведущим.
3. Реализация алгоритмов работы сопрягающего устройства:
   - функция проверки информации. Алгоритм проверки устройством входной информации на целостность частично включает в себя ранее реализованный протокол Modbus RTU;
   - создание функции информационной разгрузки ведомых устройств.

2.1. Реализация UART протокола

В процессе создания UART-протокола были выполнены этапы конфигурирования периферии и инициализация выходов STM32 в STM32CubeMX, написание алгоритма работы в Keil uVision, тестирование созданного устройства.

2.1.1. Конфигурирование периферийных устройств

Настройка функционального назначения выводов было произведено в первой из четырех рабочих вкладок STM32CubeMX (рисунок 13). Рабочая область вкладки разделена на две части: в первой отображается дерево конфигурации периферийных модулей и специализированных программных компонентов, а во второй организовано графическое представление МК и использования его выводов.

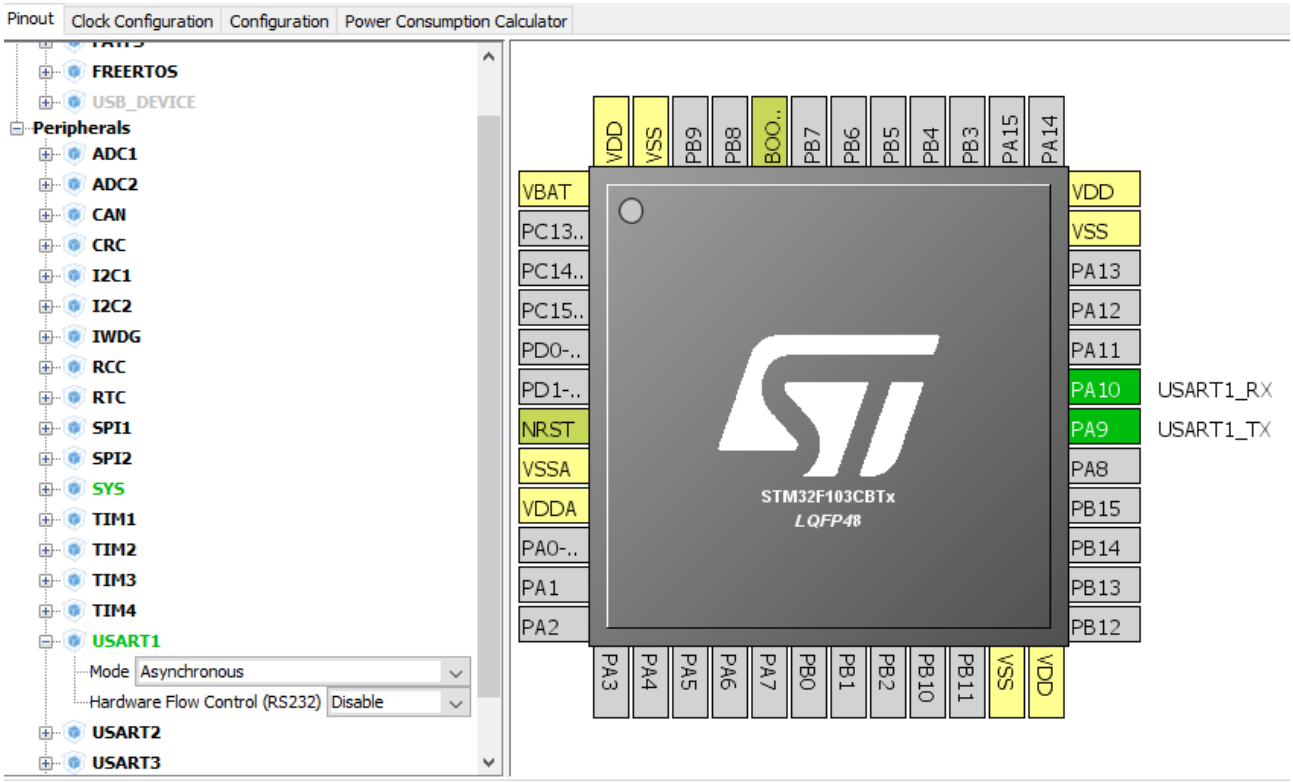


Рисунок 13 – Вкладка назначения выводов микроконтроллера

Интерфейс USART был настроен таким образом, чтобы передача данных выполнялась в асинхронном режиме.

Следующим этапом была выполнена настройка параметров UART (рисунок 14):

- скорость передачи данных в битах в секунду – 9600 бит/с;
- длина кадра – 8 бит;
- количество стоп-битов – 1;
- контроль четности – не используется;
- направление потока данных – прием и передача.

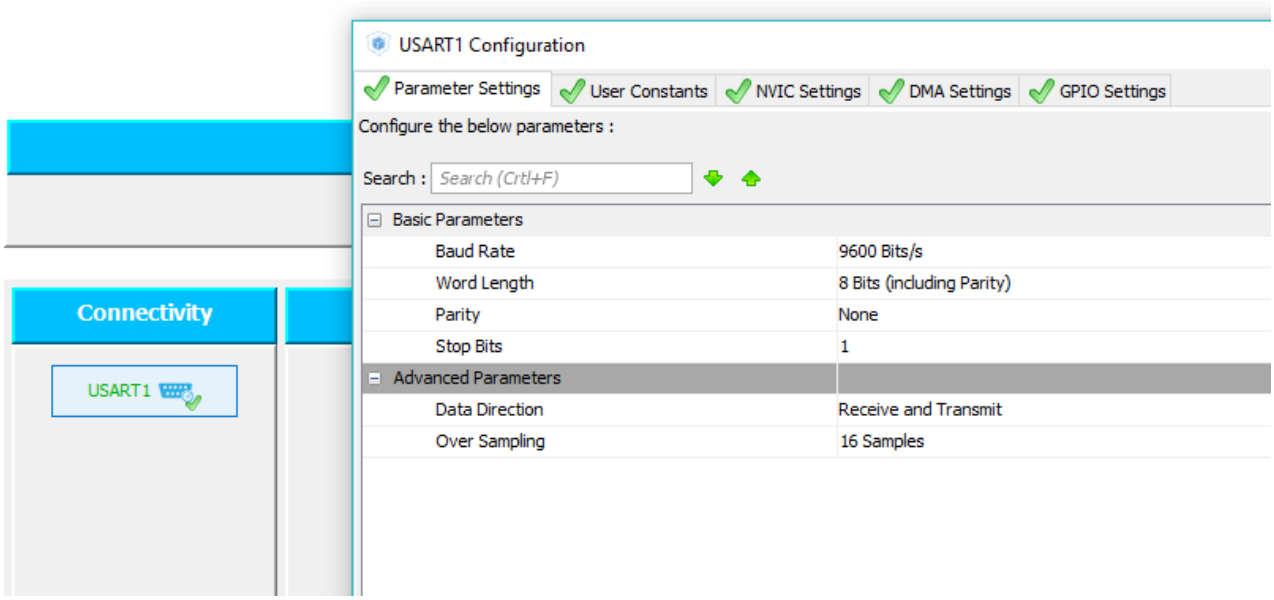


Рисунок 14 – Вкладка настройки UART

На этом необходимые настройки периферии окончены. Далее в STM32CubeMx был сгенерирован проект для среды Keil uVision, в которой был написан код основной программы.

Код программы представлен в приложении Б.

2.1.2. Алгоритм работы программы

Выполнение программы происходит в режиме активного ожидания центрального процессора, когда в цикле while процессор постоянно выполняет алгоритм, прописанный внутри цикла. Это имеет свой недостаток. Процессор постоянно ждет поступления информации по UART, даже когда информация не поступает, он все равно опрашивает входы на ее наличие. Это заметно ухудшает производительность микроконтроллера в целом, а также увеличивает риск возникновения ошибки при получении/отправке данных. В данном случае такое решение не является критичным, поскольку для стабильной работы канала связи нужна обработка только одного UART-канала с минимумом действий, но при дальнейшем создании более сложных программ целесообразнее использовать прерывания.

Созданная программа работает по следующему алгоритму:

микроконтроллер ожидает сообщения с персонального компьютера размером 1 байт. При получении сообщения, STM32 отправляет полученное сообщение обратно персональному компьютеру.

Данный алгоритм позволяет отладить функции отправки и получения данных:

- HAL_UART_Receive — функция приема фиксированного количества байт;
- HAL_UART_Transmit — функция отправки фиксированного количества байт.

2.1.3. Отладка программы

Для отладки алгоритма установлено соединение STM32 с программой Terminal v1.9b через COM-порт персонального компьютера. В терминальной программе установлены настройки UART аналогичные настройкам UART на STM32 (раздел 2.1.1).

Исходя из приведенного выше алгоритма работы программы, при отправке каждого байта информации с терминала, микроконтроллер должен отправить точно такие же данные на ПК. Для проверки написанного алгоритма используется встроенный отладчик в среде разработки Keil uVision и программа Terminal v1.9b.

В нижней части рисунка 15 показана программа Terminal v1.9b. В верхней части программы приведены настройки UART, которые идентичны настройкам UART на STM32. Ниже расположены два поля «Receive» и «Transmit». В качестве тестового сигнала был отправлен символ «7», на рисунке это показано в поле «Transmit». В поле «Receive» отображается «7», это означает, что контроллер принял семерку и отправил ее на ПК, а, следовательно, программа работает правильно.

Правильность работы алгоритма подтверждает верхнее окно «Watch 1» отладчика Keil uVision, где показана переменная str, в которую записано значение «7».

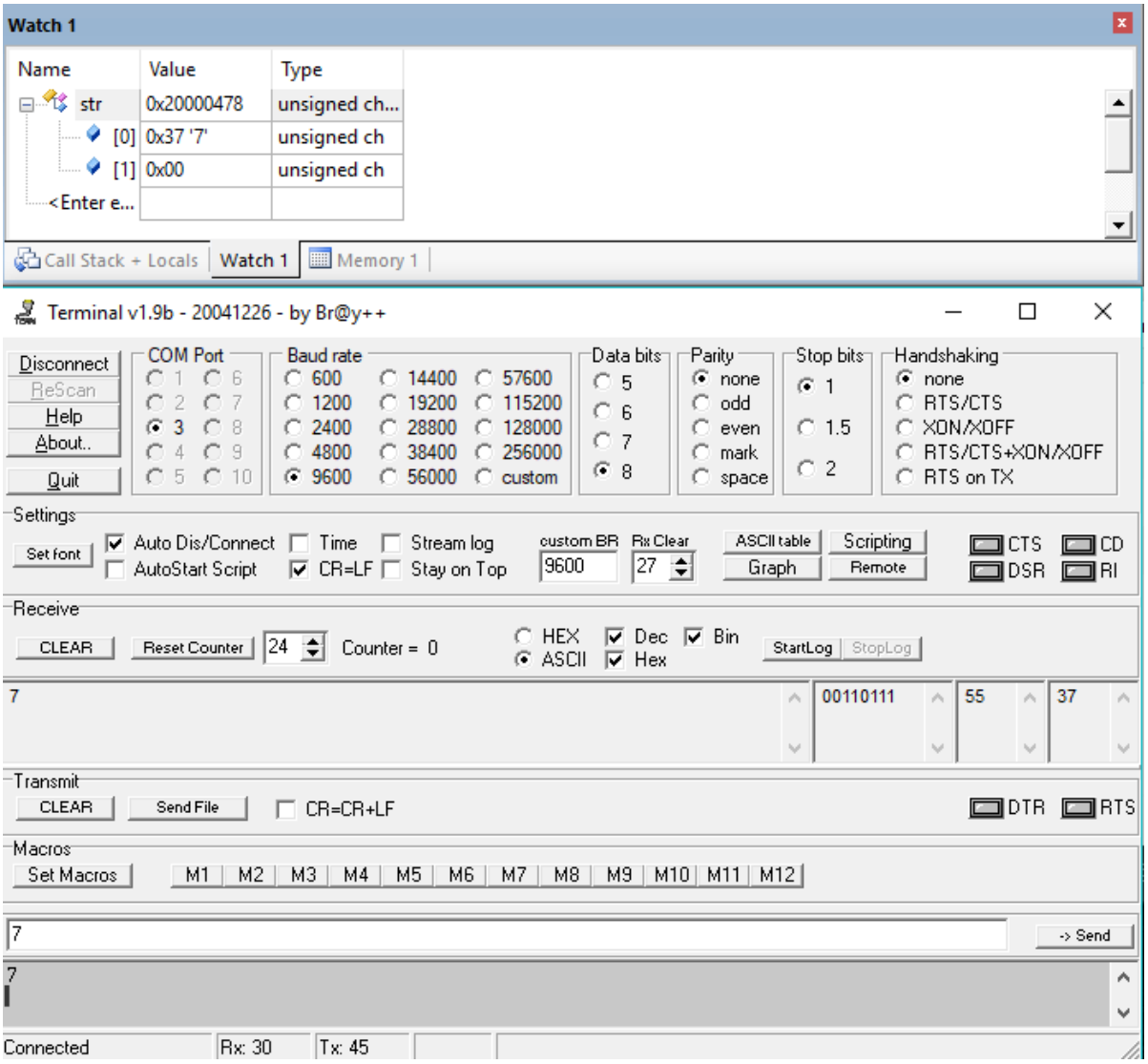


Рисунок 15 – Отладчик Keil uVision и Terminal v1.9b

2.2. Реализация протокола Modbus RTU

При создании протокола Modbus используется ранее отлаженный протокол физического уровня UART. Ведущим устройством выступает персональный компьютер, ведомым микроконтроллер.

2.2.1. Конфигурирование периферийных устройств

Конфигурация периферии микроконтроллера практически совпадает с приведенной выше конфигурацией для создания UART протокола (раздел 2.1.1). Различие заключается в том, что вся программа будет работать с использованием прерываний, а не в режиме активного ожидания. Для этого в настройке UART были разрешены прерывания (рисунок 16).

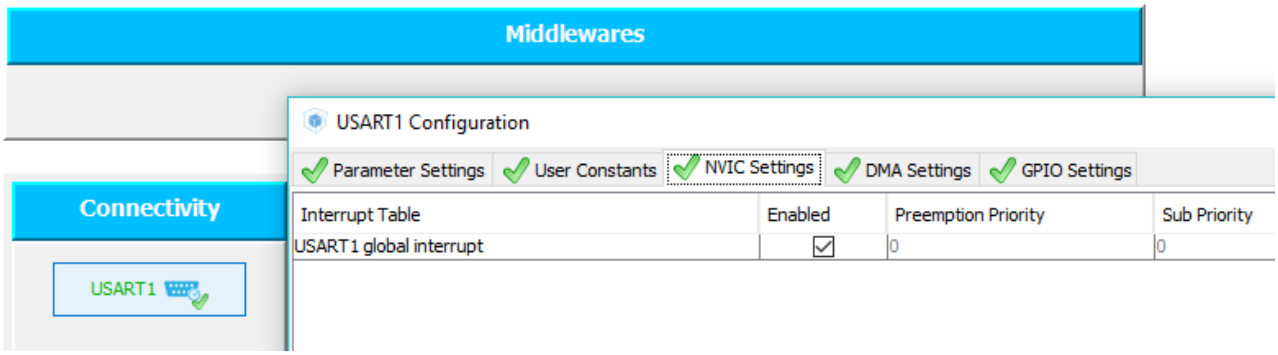


Рисунок 16 – Окно настройки прерываний UART

Прерывание - это событие, как правило, связанное с каким-либо блоком периферии микроконтроллера STM32. Событий, которые могут породить прерывание может быть множество. Например, в блоке периферии UART могут быть такие события: передача завершена, приём завершен, возникла ошибка чётности и так далее. Использование прерываний позволяет нашей программе реагировать на подобные события в режиме реального времени, а также разгружает время работы процессора, освобождая ресурсы под выполнение других задач. Сам термин прерывание означает, что прерывается выполнение основного кода программы и управление передается некоторому другому фрагменту кода, который называется обработчиком прерывания [17].

После настройки прерываний UART в STM32CubeMx был сгенерирован проект для среды Keil uVision, в которой был написан код основной программы. Код представлен в приложении В.

2.2.2. Алгоритм работы программы

На рисунках 17, 18 представлен алгоритм работы программы. Как уже было описано выше (раздел 1.3.2.2), в реализованном протоколе Modbus используются функции чтения и записи данных. Это подразумевает, что сопрягающее устройство должно уметь получать запросы разной информационной длины. Длина запроса на чтение данных составляет 8 байт, на запись данных 13. Причем устройство не может знать заранее, какой запрос отправит диспетчерская станция. Следовательно, для данной задачи невозможно использовать функцию фиксированного приема данных «HAL_UART_Receive». Данная задача была решена введением алгоритма побайтового приема данных.

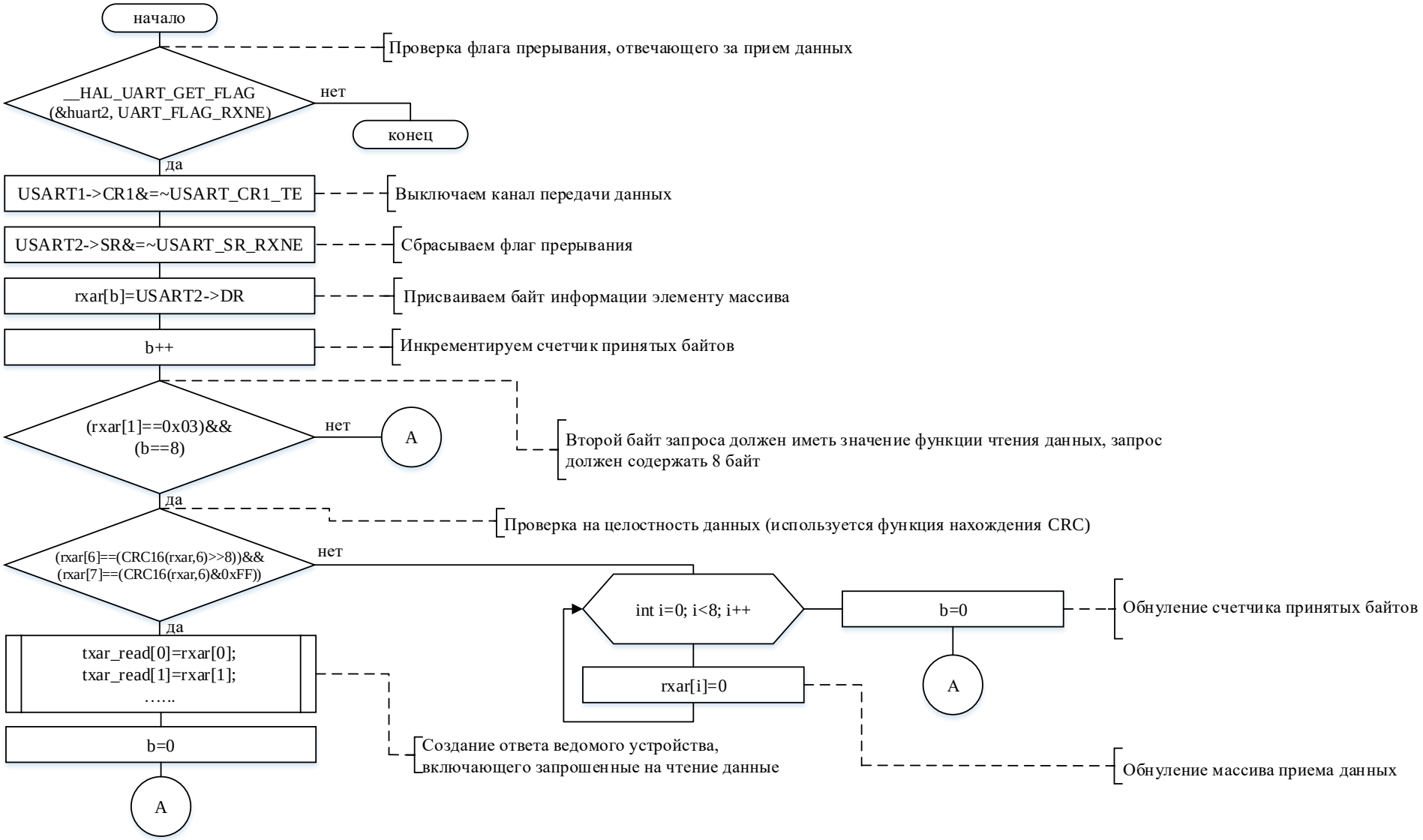


Рисунок 17 – Алгоритм приема/передачи данных по протоколу Modbus

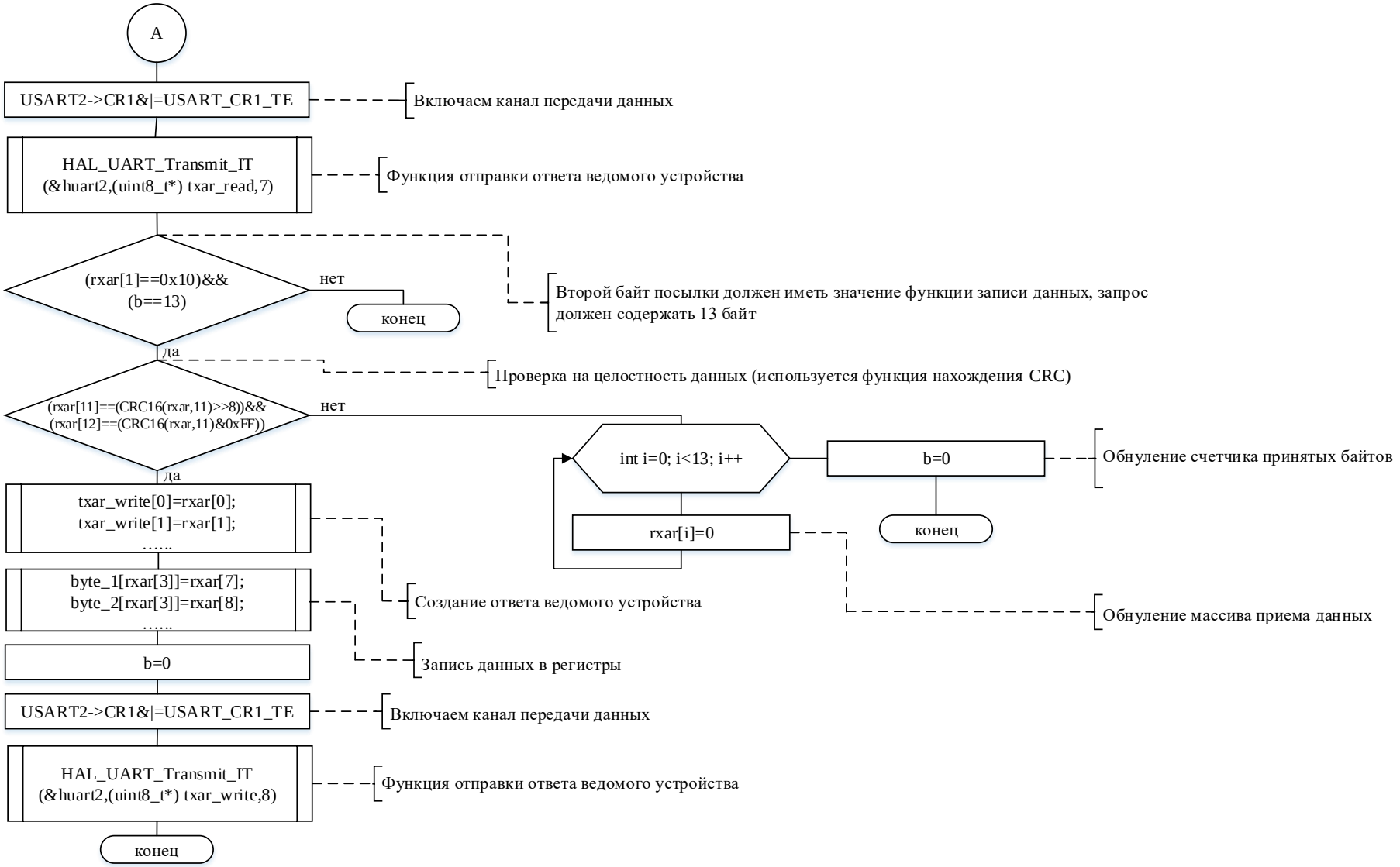


Рисунок 18 – Продолжение алгоритма приема/передачи данных по протоколу Modbus

При поступлении первого байта запроса на микроконтроллер, срабатывает прерывание по приему информации и программа начинает выполнять алгоритм, заложенный внутри обработчика этого прерывания.

Каждый принятый байт записывается в массив и инкрементируется счетчиком, то есть программа всегда знает количество полученных байтов. По правилам организации протокола Modbus, второй байт запроса идентифицирует функцию чтения или записи. Тогда микроконтроллер, обрабатывая второй байт, узнает функцию запроса, а, следовательно, узнает и всю длину запроса. Когда счетчик байтов достигает нужного значения, весь запрос обрабатывается и микроконтроллер выполняет запрос диспетчерской станции. Далее массив запроса и счетчик байтов обнуляется, после чего микроконтроллер готов принимать новый запрос.

2.2.3. Отладка программы

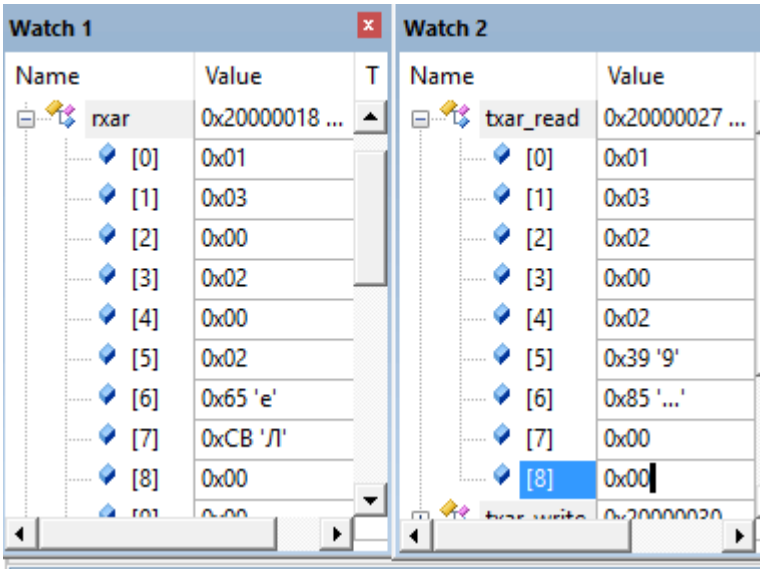
Для проверки работы протокола Modbus был создан массив для тестирования функции чтения данных — data_for_read, и массивы для тестирования функции записи данных: rec_1, rec_2, rec_3, rec_4. В таблице 6 отображено содержимое массива data_for_read.

Таблица 6 – Содержимое массива data_for_read

Индекс элемента	0	1	2	3	4	5	6	7	8	9
Значение элемента	0	1	2	3	4	5	6	7	8	9

На рисунке 19 представлена работа программы при обработке запроса диспетчерской станции на чтение данных из памяти микроконтроллера. В нижней части рисунка представлена программа Viewer, с помощью которой были запрошены данные элемента с индексом 2 массива data_for_read, данному элементу присвоено значение «2», значит, при успешном выполнении операции, в окне программы во вкладке «Значение» отобразится считанная с массива двойка. В нижней части рисунка отображены окна отладчика среды Keil uVision. В окне «Watch 1» отображен принятый микроконтроллером запрос на чтение. В окне «Watch 2» отображены данные массива txar_read, которые соответствует

формату ответа по протоколу Modbus. Первому элементу массива присвоен номер ведомого устройства, выполнившего запрос. Второму элементу код функции, которую устройство должно было выполнить. Далее идут два элемента, которых указывается количество прочитанных байтов. Элементу с индексом четыре присвоено значение, которое нужно было считать. Последним двум элементам присвоено значение посчитанного CRC. В результате программа Viewer принимает ответ микроконтроллера и записывает значение «2» в значение запрошенного параметра.

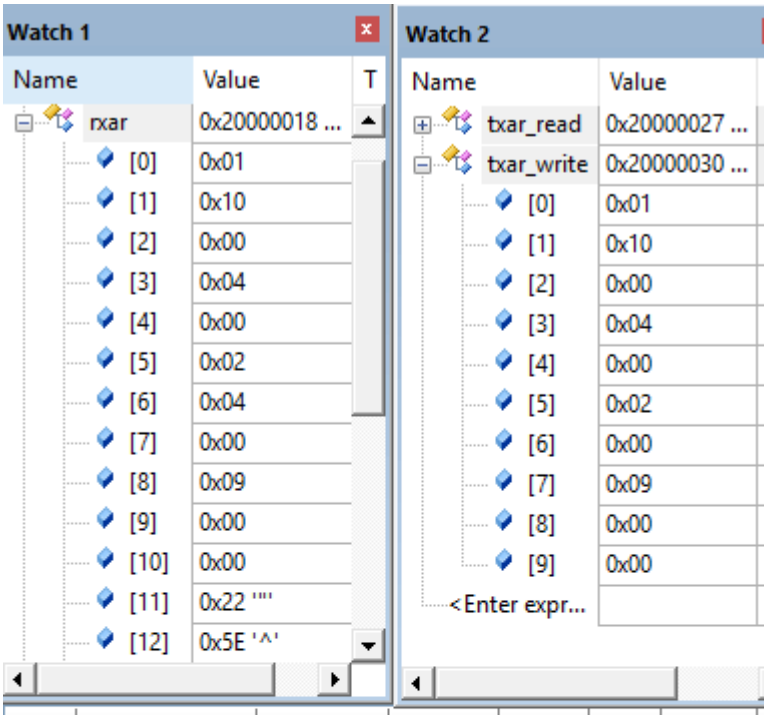


Код	Название	Значение	Ед. изм.	Тип	Вид	Адрес
A.0	test_param1			INT32	DEC	0
A.1	test_param2			INT32	DEC	1
A.2	test_param3	2		INT32	DEC	2
A.3	test_param4			INT32	DEC	3
A.4	test_param5			INT32	DEC	4
A.5	test_param6			INT32	DEC	5
A.6	test_param7			INT32	DEC	6
A.7	test_param8			INT32	DEC	7
A.8	test_param9			INT32	DEC	8
A.9	test_param10			INT32	DEC	9

Рисунок 19 – Отладка алгоритма чтения данных

На рисунке 20 представлена работа программы при обработке запроса диспетчерской станции на запись данных в память микроконтроллера. Согласно настройке формата данных для записи в программе Viewer, максимальный размер информации на запись составляет 4 байта, значит, при записи данных, в

память микроконтроллера всегда приходит 4 байта. Каждому байту соответствует отдельный массив. В нижней части рисунка, в программе Viewer значение параметра «test_param5» с адресом «4» равно девяти. Это означает, что в элемент массива микроконтроллера с индексом 4 должно быть записано число девять. Символ «9» занимает один байт памяти, следовательно, в элемент массива rec_2 с индексом четыре будет записана девятка, а в элементы массивов rec_1, rec_3 и rec_4, индекс которых равен четырем, будут записаны нули. В левой части рисунка показаны окна отладчика среды Keil uVision. В окне «Watch 1» отображен принятый микроконтроллером запрос на запись символа «9». В окне «Watch 2» отображены данные массива txar_write, которые соответствует формату ответа по протоколу Modbus. Элемент с нулевым индексом содержит номер ведомого устройства, с индексом один — код выполненной функции. Следующий элемент содержит адрес начального регистра. Элемент с индексом 3 содержит количество считанных байт. Следующие два элемента содержат количество считанных регистров. Элементы с индексами 6 и 7 содержат рассчитанный для данного ответа CRC.



Код	Название	Значение	Ед. изм.	Тип	Вид	Адрес
A.0	test_param1			INT32	DEC	0
A.1	test_param2			INT32	DEC	1
A.2	test_param3	2		INT32	DEC	2
A.3	test_param4			INT32	DEC	3
A.4	test_param5	9		INT32	DEC	4
A.5	test_param6			INT32	DEC	5
A.6	test_param7			INT32	DEC	6
A.7	test_param8			INT32	DEC	7
A.8	test_param9			INT32	DEC	8
A.9	test_param10			INT32	DEC	9

Рисунок 20 – Отладка алгоритма записи данных

Для проверки работы функции записи было проведено дополнительное тестирование. В параметр «test_param1» было записано число 32455678, а в «test_param7» число 56567. Отладчик среды Keil uVision отображает информацию на микроконтроллере в шестнадцатеричной системе исчисления. Число 32455678, записанное в десятеричной системе исчисления равно 1EF3BFE в шестнадцатеричной системе. Данное число разделено на четыре байта, каждый из которых записан в элемент с индексом 1 соответствующего массива. Число $56567_{(10)}$ равно $DCF7_{(16)}$. Это число также разделено на байты, в данном случае два байта, которые записаны в элементы с индексом 6 соответствующих массивов. На рисунке 21 отображены результаты выполнения алгоритма и показаны

данные массивов rec_1, rec_2, rec_3 и rec_4, которые подтверждают правильность выполнения программы.

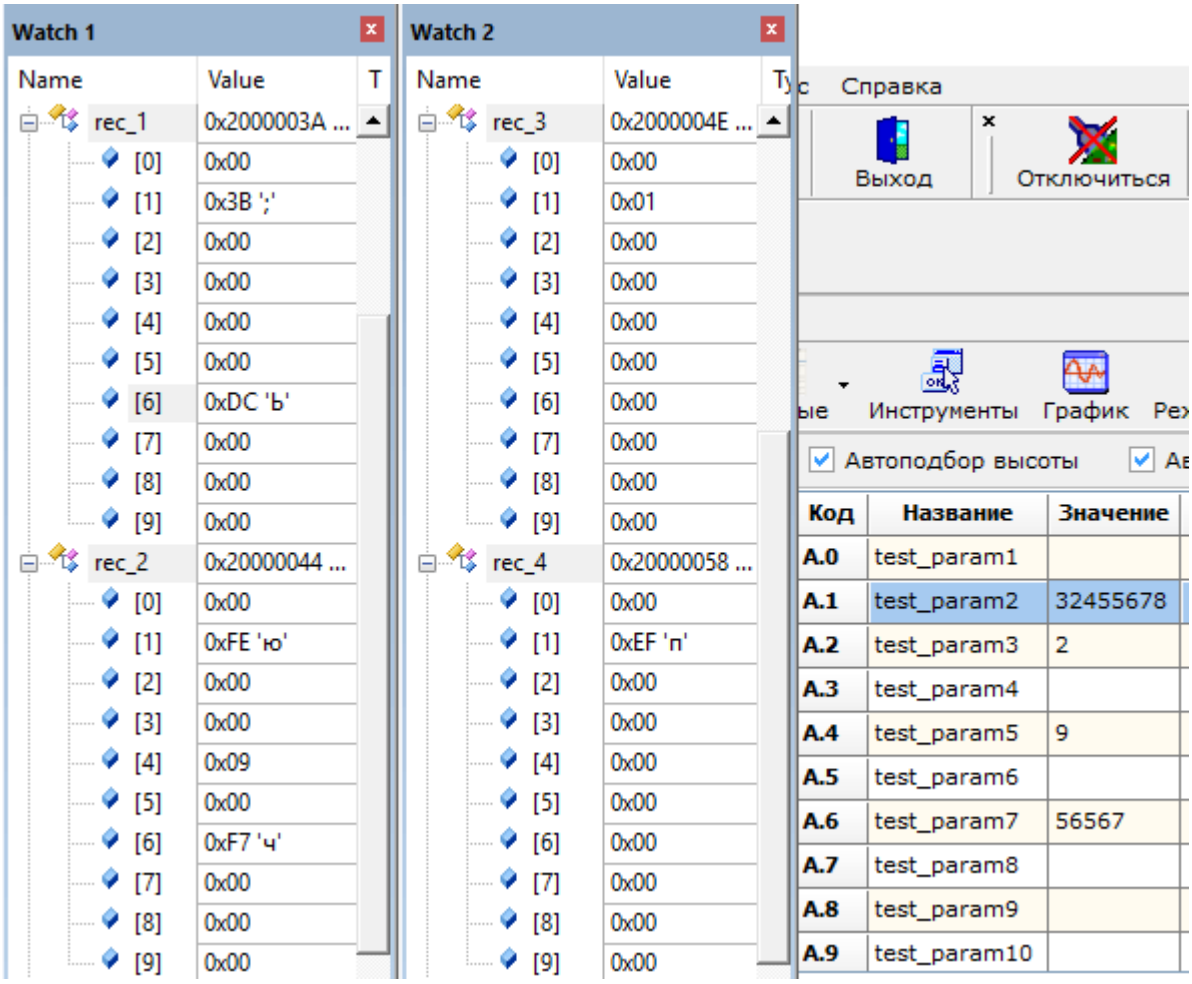


Рисунок 21 – Запись данных и массивы данных

2.3. Реализация функционала устройства сопряжения

Устройство сопряжения используется в качестве промежуточного звена между диспетчерской станцией и преобразователем частоты, которое поочередно выступает в роли ведомого и ведущего устройства. Проверяет входные данные на целостность и выполняет функцию разгрузки потока данных, поступающих на ведомые устройства. Реализация протокола Modbus на различных устройствах может отличаться по причине разной трактовки некоторых стандартов протокола. В дальнейшем планируется расширить функционал и добавить алгоритм преобразования различных входных протоколов в протокол Modbus на выходе, соответствующий протоколам подключенных ведомых устройств.

2.3.1. Конфигурирование периферийных устройств

Помимо сконфигурированного ранее канала UART, для отладки работы устройства задействовали два дополнительных UART канала (рисунок 22) и задали им параметры, аналогичные первому каналу.

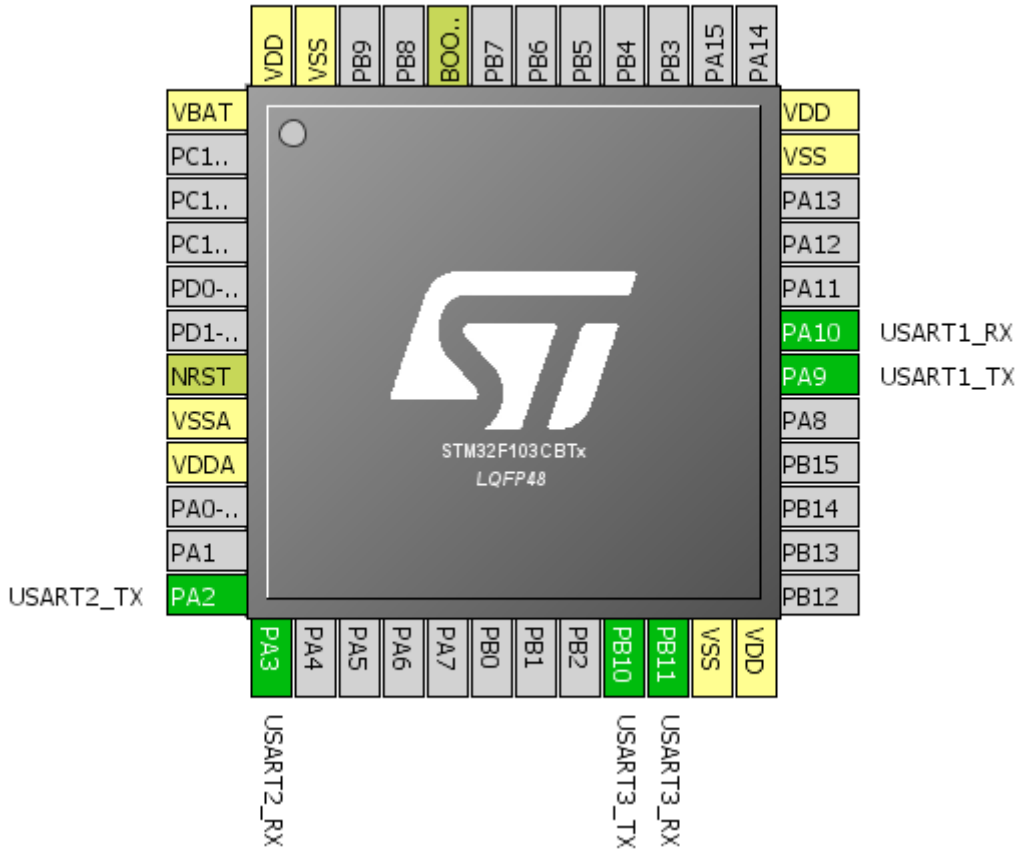


Рисунок 22 – Конфигурация трех UART каналов

Далее STM32CubeMx был сгенерирован проект для среды Keil uVision, в которой был написан код основной программы.

Код программы представлен в приложении Г.

2.3.2. Алгоритм работы устройства

На рисунках 23, 24 представлен алгоритм работы устройства. Данный алгоритм использует схожую последовательность действий, с ранее созданным алгоритмом обмена данными по протоколу Modbus. Отличия алгоритма заключаются в том, что сопрягающее устройство не создает ответ для диспетчерской станции, а пересылает его от подключенного к нему ведомого устройства. На разных UART-каналах устройство ожидает разные форматы ответов.

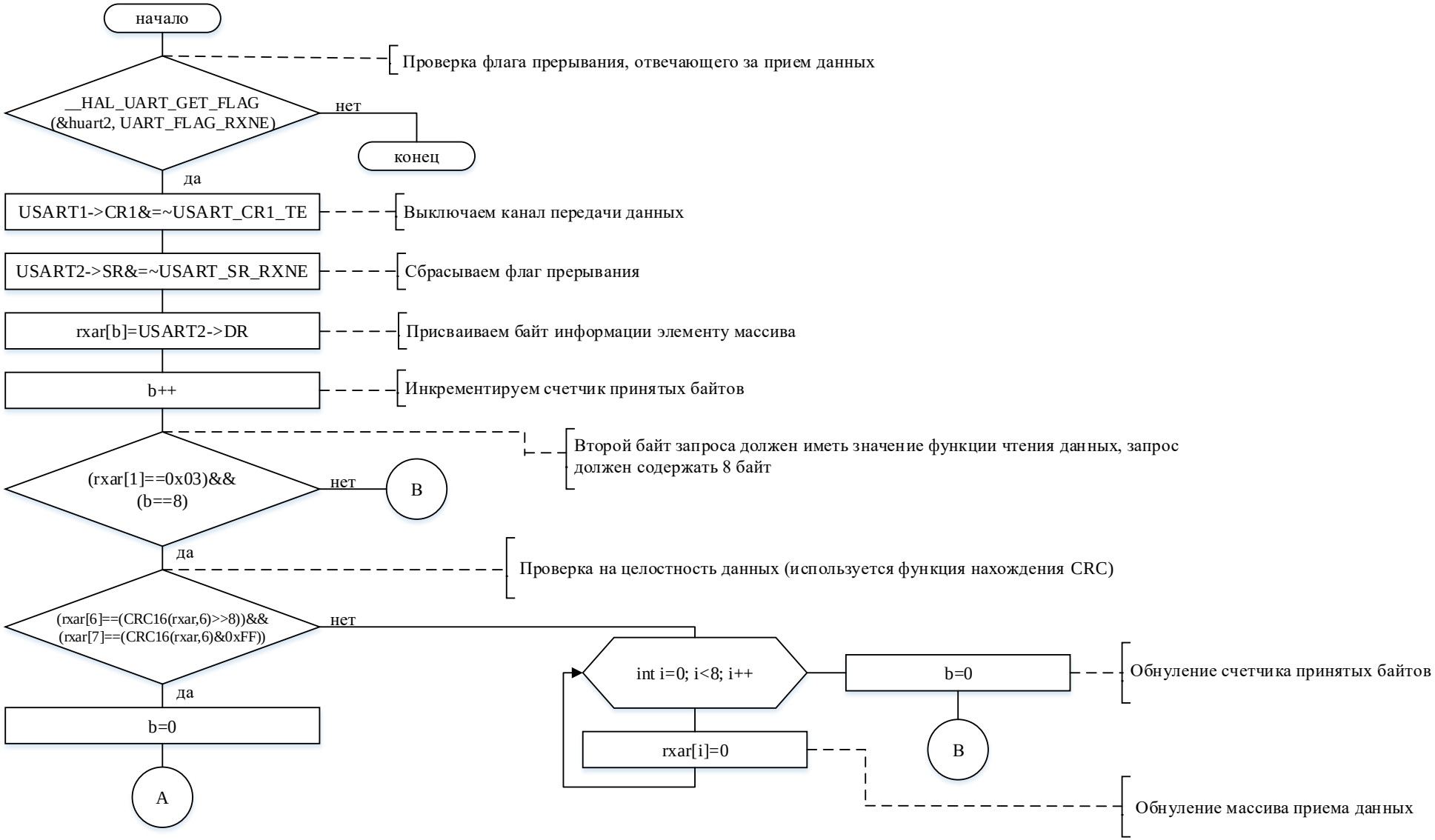


Рисунок 23 – Алгоритм работы сопрягающего устройства

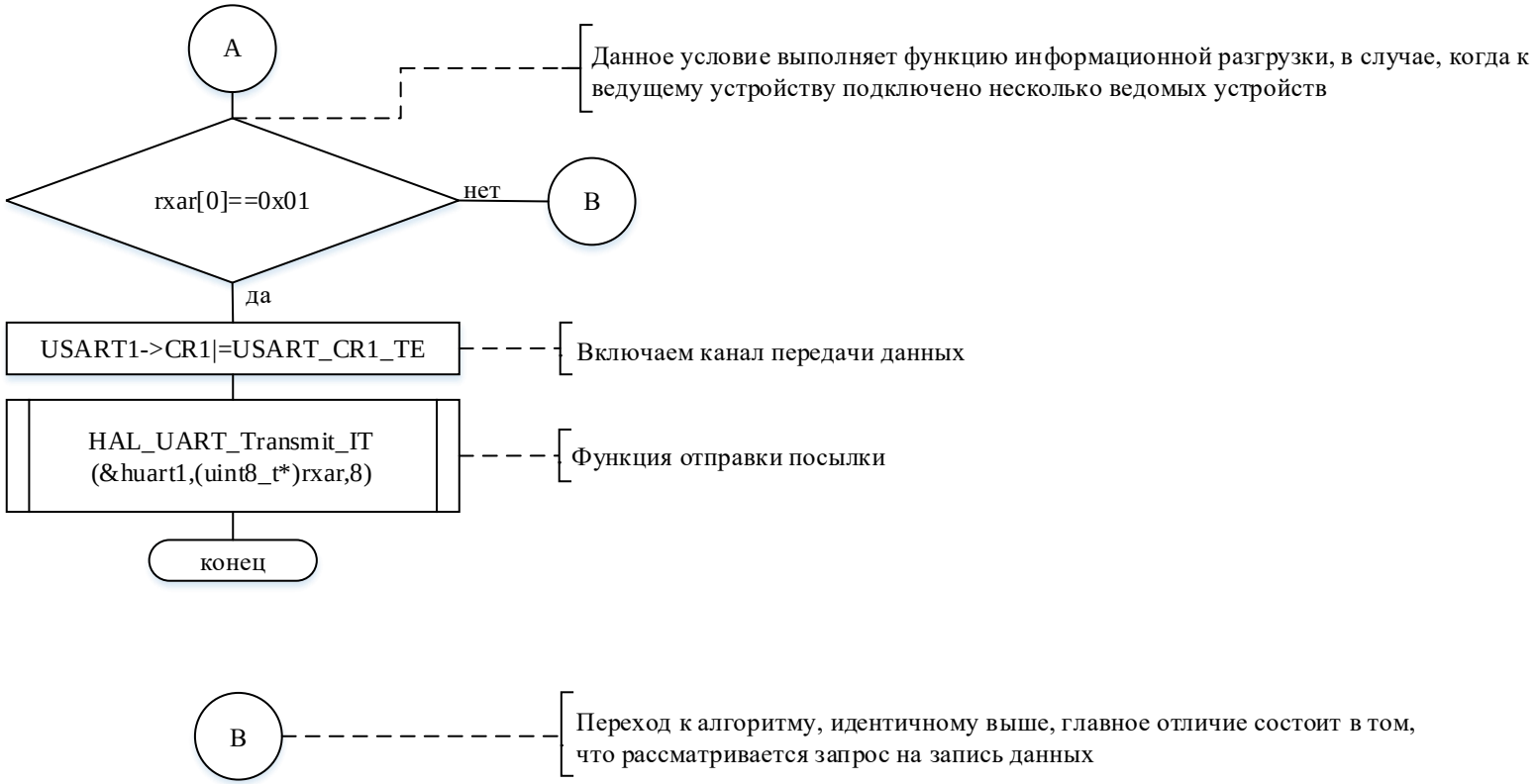


Рисунок 24 – Продолжение алгоритма работы сопрягающего устройства

В разделе 1.3.2 было показано, что устройства, обменивающиеся данными по протоколу Modbus, используют топологию «Общей шины». То есть ведущее устройство отправляет команду всем ведомым устройствам. Каждое ведомое устройство обязано сверить адрес полученной команды со своим и, в случае совпадения адресов, выполнить команду. Тем самым, при отправке любой команды с ведущего устройства, все ведомые устройства выполняют проверку этой команды на соответствие адреса.

Для уменьшения информационного потока на ведомые устройства, в STM32 была создана функция, осуществляющая селекцию ведомых устройств в зависимости от адреса, указанного в команде ведущего устройства. Иными словами, созданная функция разгружает работу ведомых устройств и гарантирует, что на каждое ведомое устройство придет команда, предназначенная именно ему.

Сопоставление адресов в командах и каналов, подключенных ведомых устройств производится в теле программы микроконтроллера с помощью оператора ветвления CASE. Структурная схема алгоритма представлена на рисунке 25.

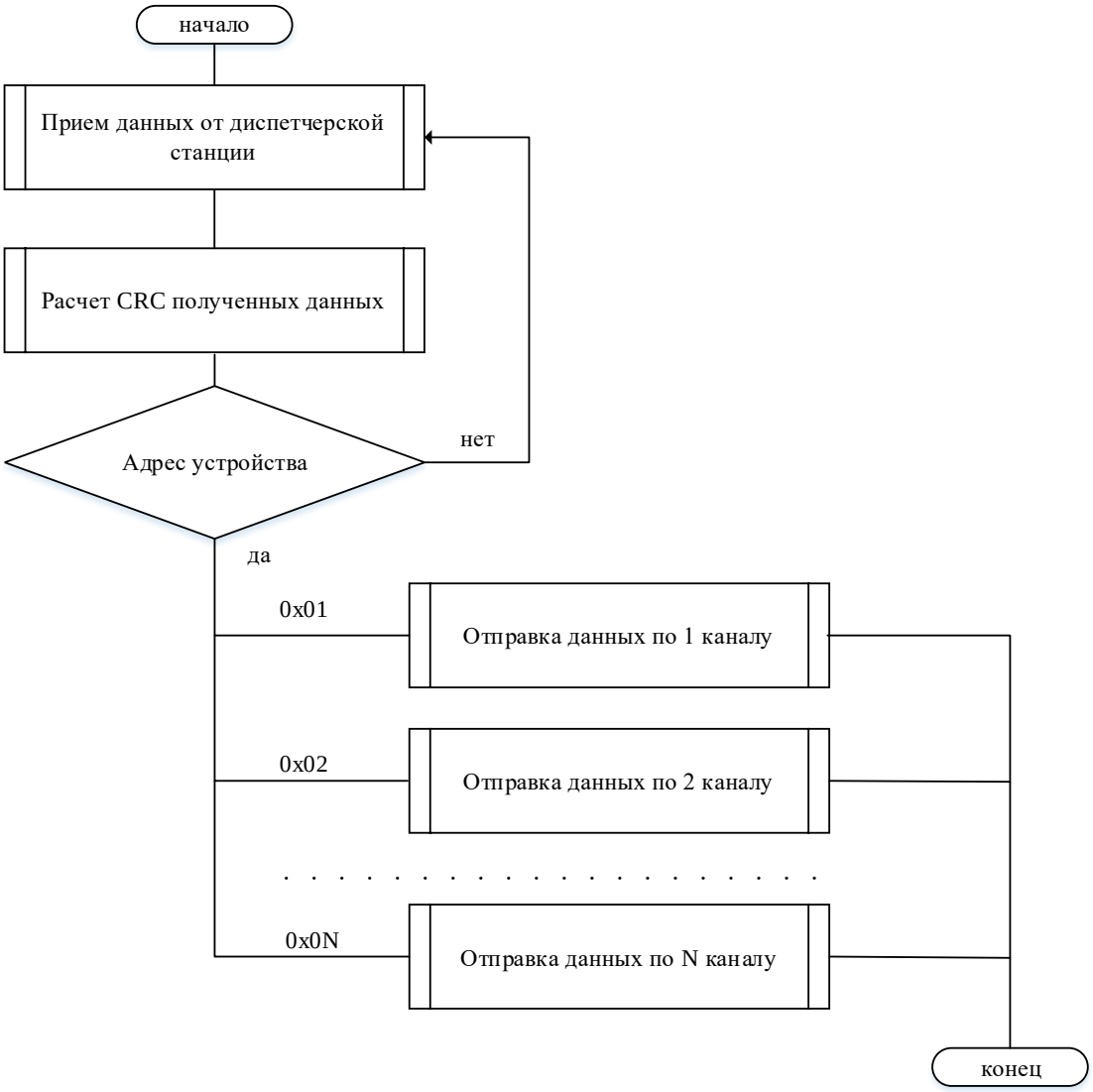


Рисунок 25 – Алгоритм разгрузки информационного потока

2.3.3. Отладка сопрягающего устройства

Отладка устройства производилась без подключения преобразователя частоты, поэтому для имитации отправки команд на ведомое устройство и имитации получения ответов использовались три UART канала.

Данные с диспетчерской станции на сопрягающее устройство поступают по каналу UART2, после проверки данных на целостность он ретранслируются на ведомое устройство по каналу UART1. Для тестирования алгоритма канал UART1 физически подключен к каналу UART3, значит, чтобы узнать, что данные по каналу UART1 успешно ретранслировались, необходимо считать полученные данные на канале UART3. В случае совпадения данных, отправленных с диспетчерской станции и данных, полученных на канале UART3, можно

утверждать, что ведомое устройство, подключенное к сопрягающему по каналу UART1, успешно получило команду от диспетчерской станции.

На рисунке 26 показаны окна отладчика среды Keil uVision. В окне Watch 1 показана информация, записанная в массив «rxar», которую получило устройство сопряжения от диспетчерской станции по каналу UART2. В окне Watch 2 отображена информация, записанная в массив rxar3, полученная по каналу UART3.

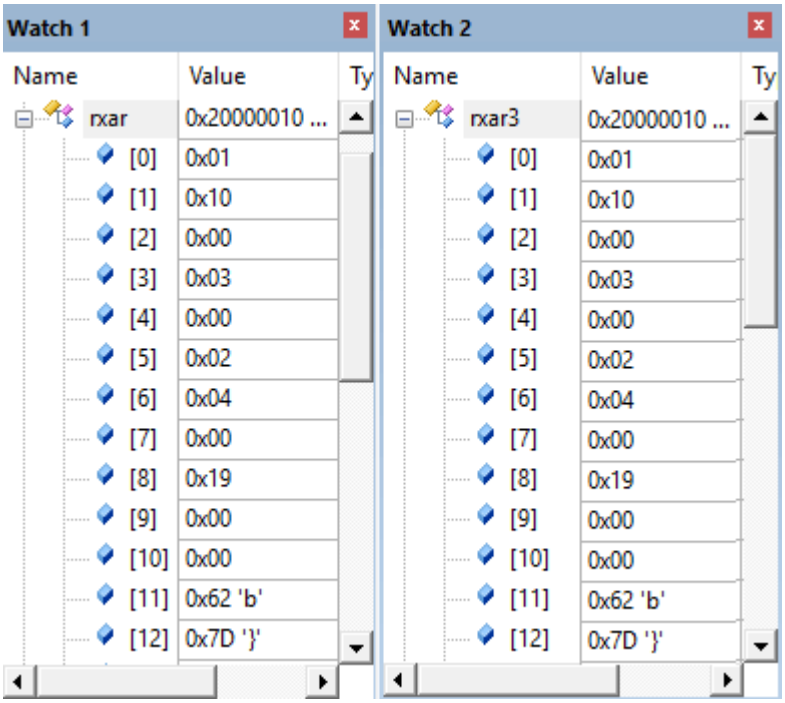


Рисунок 26 – Данные массивов «rxar» и «rxar3»

Содержимое массивов на обоих окнах идентично, следовательно, алгоритм работает верно.

Процесс приема информации с ведомого устройства и ее ретрансляции на диспетчерскую станцию по своему алгоритму схож с приведенным выше, поэтому в отладке данного процесса меняется только последовательность задействования UART-каналов. Для проверки алгоритма было проведено аналогичное тестирование, которое показало успешное выполнение заданного алгоритма. Следовательно, созданное сопрягающее устройство работает согласно поставленному заданию.

2.4. Выводы по главе 2

Во второй главе были рассмотрены этапы создания сопрягающего устройства. На физическом уровне был организован протокол UART и проведено его тестирование, которое показало правильность работы алгоритма. На канальном и прикладном уровне передачи данных был создан протокол Modbus RTU. Тестирование алгоритмов передачи данных по протоколу Modbus показало успешные результаты. Заключительным этапом было создание устройство, в алгоритмы работы которого вошли оба, ранее созданных и отлаженных, протокола, а также функция разгрузки входной информации на ведомые устройства. Поэтапная разработка и тестирование алгоритмов позволили успешно создать итоговое сопрягающее устройство.

3. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

В настоящее время оценка коммерческой разработки является важным условием для поиска источников финансирования. Нужно уметь оценить свои силы на рынке конкурентов, увидеть недостатки в их продукции и минимизировать их в своем проекте, оценить преимущества и индивидуальность своего проекта для поиска источников финансирования. Коммерциализация результатов проекта необходима разработчикам, так как способствует дальнейшему обеспечению развития проекта и его доведению до конечного продукта.

3.1. Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения

3.1.1. Потенциальные потребители результатов исследования

Потенциальными потребителями данного проекта могут быть различные компании, предприятия и фирмы, ведущие разработки в сферах автоматизации и связи.

В приложении Д представлена карта сегментации рынка, в которой учитываются потребности разных по величине предприятий в использовании управляющих устройств. Опираясь на данную карту можно сделать выводы:

1. Промышленные контроллеры используют крупные предприятия, это связано с тем, что, в условиях сложных и опасных технологических процессов, большое внимание уделяется надежности системы. Промышленные контроллеры могут изготавливается во взрывозащищенных корпусах и обладают высоким классом надежности, однако, обладают высокой стоимостью, и поэтому часто недоступны для мелких и средних предприятий.

2. Средние предприятия могут использовать контроллеры разных типов, в зависимости от вида технологического процесса и требуемых показателей надежности и качества процесса. Во многом такие предприятия первыми пробуют внедрять микроконтроллеры в новые сферы человеческой деятельно-

сти. Благодаря таким компаниям и фирмам растет популярность среди микроконтроллеров STM32 и других.

3. Мелкие предприятия, проекты которых включают в себя «домашнюю» автоматизацию, создание интернет-вещей, широко используют микроконтроллеры различных производителей. Для начального изучения и освоения микроконтроллеров хорошо подходят устройства на базе Arduino, обладающие широким выбором дополнительных устройств управления, датчиков и исполнительных механизмов.

3.1.2. Анализ конкурентных технических решений

Детальный анализ конкурирующих разработок помогает вносить коррективы в научное исследование, чтобы успешнее противостоять своим соперникам. Важно реалистично оценить сильные и слабые стороны разработок конкурентов.

С этой целью может быть использована вся имеющаяся информация о конкурентных разработках:

технические характеристики разработки;

конкурентоспособность разработки;

уровень завершенности научного исследования (наличие макета прототипа и т.п.);

бюджет разработки;

уровень проникновения на рынок;

финансовое положение конкурентов, тенденции его изменения и т.д.

Анализ конкурентных технических решений с позиции ресурсоэффективности и ресурсосбережения позволяет провести оценку сравнительной эффективности научной разработки и определить направления для ее будущего повышения. В таблице 7 представлена оценочная карта для сравнения конкурентных технических решений.

Таблица 7 – Оценочная карта для сравнения конкурентных технических решений (разработок)

Критерии оценки	Вес критерия	Баллы			Конкуренто-способность		
		$Б_ф$	$Б_{к1}$	$Б_{к2}$	$К_ф$	$К_{к1}$	$К_{к2}$
1	2	3	4	5	6	7	8
Технические критерии оценки ресурсоэффективности							
1. Надежность	0,15	3	4	5	45	0,6	0,75
2. Масштаби-руемость	0,1	5	5	5	0,5	0,5	0,5
3. Возможно-сти продукта	0,15	5	4	4	0,75	0,6	0,6
4. Удобство в эксплуатации	0,1	4	3	4	0,4	0,3	0,4
5. Качество интерфейса пользователя	0,15	4	5	4	0,6	0,75	0,6
Экономические критерии оценки эффективности							
1. Конкурен-тоспособность продукта	0,1	5	4	4	0,5	0,4	0,4
2. Цена про-дукта	0,1	5	3	3	0,5	0,3	0,3
3. Сервисное обслуживание	0,1	4	4	4	0,4	0,4	0,4
4. Предпола-гаемый срок эксплуатации	0,05	5	5	5	0,25	0,25	0,25
Итого	1				4,35	4,1	4,2

Для оценки ресурсоэффективности были выбраны следующие критерии: надежность, масштабируемость, возможности продукта, удобство в эксплуатации и качество интерфейса пользователя. Каждый из критериев играет важную роль при выборе продукта. На любом производстве важна степень надежности используемого оборудования, по данному критерию продукт проигрывает конкурентам, но имеет все возможности на повышение надежности при дальнейших разработках. По остальным техническим показателям продукт является конкурентно способным. Сильными сторонами продукта являются масштабируемость и возможность использования в различных сферах производства.

Для оценки экономической эффективности были выбраны следующие экономические критерии: конкурентоспособность продукта, цена, сервисное обслуживание, предполагаемый срок эксплуатации.

Результаты анализа выявили, что созданный продукт имеет низкую себестоимость, в сравнении с конкурентными продуктами и, при развитии технологии, обладает большим потенциалом для создания и развития новых продуктов.

Также целью является не только продажа продукта, но и дальнейшее послепродажное обслуживание. Помощь при внедрении в эксплуатацию и сопровождение продукта поможет заинтересовать и завоевать доверие покупателей.

3.1.3. SWOT-анализ

SWOT – Strengths (сильные стороны), Weaknesses (слабые стороны), Opportunities (возможности) и Threats (угрозы) – представляет собой комплексный анализ научно-исследовательского проекта. SWOT-анализ применяют для исследования внешней и внутренней среды проекта.

Первый этап заключается в описании сильных и слабых сторон проекта, в выявлении возможностей и угроз для реализации проекта, которые проявились или могут появиться в его внешней среде.

Сильные стороны – это ресурсы или возможности, которыми располагает руководство проекта и которые могут быть эффективно использованы для достижения поставленных целей.

Слабые стороны – это то, что плохо получается в рамках проекта или где он располагает недостаточными возможностями или ресурсами по сравнению с конкурентами.

Возможности включают в себя любую предпочтительную ситуацию в настоящем или будущем, возникающую в условиях окружающей среды проекта.

Угроза представляет собой любую нежелательную ситуацию, тенденцию или изменение в условиях окружающей среды проекта, которые имеют разрушительный или угрожающий характер для его конкурентоспособности в настоящем или будущем.

Таблица 8 – матрица SWOT

	Сильные стороны: С1. Низкая цена. С2. Широкий спектр сфер применения. С3. Возможность расширения и дополнения проекта. С4. Обновление и совершенствование комплектующих проекта.	Слабые стороны: Сл1. Надежность Сл2. Среды программирования Сл3. Время разработки
Возможности: В1. Повышение надежности комплектующих проекта В2. Осваивание новых отраслей. В3. Увеличение масштаба проектов. В4. Повышение квалификации сотрудников.		
Угрозы: У1. Ужесточение конкуренции У2. Подорожание ресурсов. У3. Экономическая нестабильность.		

Анализ интерактивных таблиц представляется в форме записи сильно коррелирующих сильных сторон и возможностей, или слабых сторон и возможностей и т.д. В случае, когда две или более возможности сильно коррелируют с одними и теми же сильными сторонами, с большой вероятностью можно говорить об их единой природе.

Таблица 9 – Интерактивная матрица проекта (сильные стороны)

	С1	С2	С3	С4
В1	-	-	-	+
В2	+	+	+	-
В3	+	+	+	-
В4	+	+	+	-

	С1	С2	С3	С4
У1	+	-	-	+
У2	-	+	+	-
У3	-	+	+	-

Таблица 10 – Интерактивная матрица проекта (слабые стороны)

	Сл1	Сл2	Сл3
В1	+	-	-
В2	-	+	-
В3	-	+	-
В4	-	-	+
У1	+	+	-
У2	+	+	-
У3	-	-	+

Составив и проанализировав интерактивную матрицу проекта, составим итоговую матрицу SWOT-анализа.

Таблица 11 – Полная матрица SWOT

	Сильные стороны: С1. Низкая цена. С2. Широкий спектр сфер применения. С3. Возможность расширения и дополнения проекта. С4. Обновление и совершенствование комплектующих проекта.	Слабые стороны: Сл1. Надежность Сл2. Среды программирования Сл3. Время разработки
Возможности: В1. Повышение надежности комплектующих проекта В2. Осваивание новых отраслей. В3. Увеличение масштаба проектов.	В1С4 Повышение надежности при дальнейшем совершенствовании продукции В2В3В4С1С2С3 Увеличение масштаба проектов и расширение сфер применения за счет низких цен, вследствие, рост квалификации сотрудников и рабочих	В1Сл1 Повышение показателей надежности в результате новых разработок В2В3Сл2 В результате увеличения масштабов эксплуатации разработка более удобного и адаптивного ПО В4Сл3

В4. Повышение квалификации сотрудников.	мест	Уменьшение времени разработки за счет квалифицированного персонала
Угрозы: У1. Ужесточение конкуренции У2. Подорожание ресурсов. У3. Экономическая нестабильность.	У1С1С4 Снижение стоимости за счет совершенствования и оптимизации продукта У2У3С2С3 Устойчивость к экономической нестабильности и удержание клиентов с счет широкой сферы применения продукта и его дополнительных возможностей.	У1У2Сл1Сл2 Мониторинг рынка конкурентов, освоение конкурентных решений У3Сл3 Просчет временных издержек на поставку и разработку продукта

3.2. Планирование проектных работ

3.2.1. Структура работ в рамках проекта

Планирование комплекса предполагаемых работ осуществлено в следующем порядке:

- определение структуры работ проекта;
- определение участников каждой работы;
- установление продолжительности работ;
- построение графика проведения проектной работы.

Для выполнения технического задания была сформирована рабочая группа. По каждому виду запланированных работ установлена соответствующая должность исполнителей.

В данном разделе составлен перечень этапов и работ проекта, а также, в таблице 12 произведено распределение исполнителей по видам работ.

Таблица 12 – Перечень этапов, работ и распределение исполнителей

Основные этапы	№ раб	Содержание работ	Должность исполнителя
Разработка технического задания	1	Составление и утверждение технического задания	Студент, Руководитель
Разработка про-	2	Календарное планирование ра-	Студент, Ру-

Основные этапы	№ раб	Содержание работ	Должность исполнителя
граммы выполнения работы		бот	ководитель
Изучение среды разработки микроконтроллера	3	Изучение возможного ПО для программирования STM32	Студент
	4	Выбор наиболее подходящего ПО и его углубленное изучение	Студент
Создание и отладка связи между ПК и STM32	5	Отправка и прием информации на STM32, используя интерфейс передачи данных USART	Студент
	6	Тестирование работы при разных скоростях передачи	
Изучение протокола Modbus RTU	7	Изучение правил приема передачи данных	Студент
	8	Изучение табличного способа расчета CRC	
	9	Изучение функции чтения и функции записи протокола Modbus RTU	
Реализация протокола Modbus RTU	10	Программирование протокола и его отладка	Студент
Реализация связывающего устройства на базе STM32 между ПК и ЧП	11	Создание функции выбора устройства на STM32	Студент
	12	Отладка работы устройства, запуск частотного преобразователя	
Изложение выполненной работы в пояснительной записке	13	Создание отчета по проектной работе	Студент, Руководитель

3.2.2. Определение трудоемкости выполнения работ

Трудовые затраты в большинстве случаев образуют основную часть стоимости разработки, поэтому важным моментом является определение трудоемкости работ каждого из участников проекта.

Трудоемкость выполнения проекта оценивается экспертным путем в человеко-днях и носит вероятностный характер, т.к. зависит от множества трудно учитываемых факторов. Для определения ожидаемого (среднего) значения трудоемкости $t_{ожi}$ используется следующая формула:

$$T_{ожі} = \frac{3t_{mini} + 2t_{maxi}}{5}, \qquad (1)$$

где $t_{ожі}$ – ожидаемая трудоёмкость выполнения i-ой работы, человеко-дни;

t_{mini} – минимально возможная трудоемкость выполнения заданной i-ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств), человеко-дни;

t_{maxi} – максимально возможная трудоемкость выполнения заданной i-ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств), человеко-дни.

По формуле 2, рассчитывается продолжительность каждой работы в рабочих днях T_{pi}, с учетом численности исполнителей на каждом этапе выполнения работ.

$$T_{p_i} = \frac{t_{ожі}}{Ч_i}, \quad (2)$$

где T_{p_i} – продолжительность i-ой работы, рабочие дни;

$t_{ожі}$ – ожидаемая трудоёмкость выполнения i-ой работы, человеко-дни;

$Ч_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, человек.

Для удобства построения графика проведения научного исследования необходимо перевести длительность каждого из этапов работ из рабочих в календарные дни с помощью формулы 3.

$$T_{кі} = T_{pi} \cdot k_{кал}, \quad (3)$$

где $T_{кі}$ – продолжительность выполнения i-й работы в календарных днях;

T_{pi} – продолжительность выполнения i-й работы в рабочих днях;

$k_{кал}$ – коэффициент календарности.

Для расчёта длительности каждого из этапов работ в календарных днях необходимо рассчитать коэффициент календарности $k_{кал}$ используя формулу 4.

$$K_{кал} = \frac{T_{кал}}{T_{кал} - T_{вых} - T_{пр}}, \quad (4)$$

где $k_{кал}$ – коэффициент календарности;

$Т_{кал}$ – количество календарных дней в году;

$Т_{кал}$ – количество выходных дней в году;

$Т_{пр}$ – количество праздничных дней в году.

В 2018 году количество календарных дней составляет 365 дней, а сумма выходных и праздничных дней равна 118 дням. Из этого следует, что коэффициент календарности для 2018 года равен $k_{кал} = 1{,}478$.

Для построения календарного плана-графика необходимо рассчитать временные показатели проведения научного исследования. Все расчеты представлены в таблице 13.

Таблица 13 – Временные показатели проведения научного исследования

Название работы	Исполнители	Трудоемкость работ, человеко-дни			Длительность работ	
		t_{min}	t_{max}	$t_{ожі}$	$T_р$, рабочие дни	$T_к$, календарные дни
Составление и утверждение технического задания	2	2	5	3	3	4
Календарное планирование работ	2	1	2	1	1	1
Изучение возможного ПО для программирования STM32	1	2	3	2	2	3
Выбор наиболее подходящего ПО и его углубленное изучение	1	5	7	5	5	6
Отправка и прием информации на STM32, используя интерфейс передачи данных USART	1	10	25	16	16	18
Тестирование работы при разных скоростях передачи	1	4	6	6	6	13
Изучение правил приема передачи данных	1	4	8	4	4	7
Изучение табличного способа расчета CRC	1	1	3	2	2	3
Изучение функции чтения и функции записи протокола Modbus RTU	1	5	15	10	10	15
Программирование протокола и его отладка	1	15	45	15	15	20
Создание функции выбора устройства на STM32	1	5	10	10	10	13
Отладка работы устройства, запуск частотного преобразователя	1	10	25	12	12	18
Создание отчета по проектной работе	2	4	6	5	5	7
Итого				91	91	128

3.2.3. Разработка графика проведения научного исследования

Для наглядного представления распределения работ участников проекта и затраченного времени была построена диаграмма Ганта – горизонтальный ленточный график, на котором работы по теме представляются протяженными во времени отрезками, характеризующимися датами начала и окончания выполнения данных работ. Построенная диаграмма Ганта представлена в приложении Б.

3.3. Бюджет научно-технического исследования

3.3.1. Основная заработная плата исполнителей темы

Расчеты затрат на основную заработную плату приведены в таблице 14. При расчете учитывалось, что в 2018 году 247 рабочих дней. Основная заработная плата (ЗП) сотрудника от предприятия рассчитывается по следующей формуле 5:

$$З_{осн} = З_{дн} * Т_р, \qquad (5)$$

где $З_{осн}$ – основная ЗП одного работника;

$Т_р$ – продолжительность работ, выполняемых научно-техническим работником (раб. дн.);

$З_{дн}$ – среднедневная ЗП (руб.)

Среднедневная ЗП рассчитывается по формуле 6:

$$З_{дн} = \frac{З_М * М}{F_Д}, \qquad (6)$$

где $З_М$ – месячный должностной оклад работника, руб.;

М – количество месяцев работы без отпуска в течение года:

при отпуске в 24 раб. дня М =11,2 месяца, 5-дневная неделя;

$F_Д$ – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн. (247 рабочих дней в 2018 году).

Месячный должностной оклад работника:

$$З_М = З_{ОК} * k_p, \qquad (7)$$

где $З_{ОК}$ – оклад (руб.); k_p– районный коэффициент, равный 1,3 (для Томска).

Таблица 14 – расчёт основной ЗП

Исполнитель	$З_{ОК}$, руб.	k_p	$З_{дн}$, руб.	$Т_р$, раб. дн.	$З_{осн}$, руб.
НР	28000	1,3	1272	12	15264
И	17000	1,3	769	128	98432
Итого:					113696

Затраты на заработную плату без учета районного коэффициента равны 87,458 руб.

3.3.2. Дополнительная заработная плата исполнительной

Затраты по дополнительной заработной плате исполнителей темы учитывают величину предусмотренных Трудовым кодексом РФ доплат за отклонение от нормальных условий труда, а также выплат, связанных с обеспечением гарантий и компенсаций (при исполнении государственных и общественных обязанностей, при совмещении работы с обучением, при предоставлении ежегодного оплачиваемого отпуска и т.д.).

Расчет дополнительной заработной платы ведется по следующей формуле 8:

$$З_{доп} = k_{доп} * З_{осн}, \qquad (8)$$

где $k_{доп}$ – коэффициент дополнительной заработной платы (0,12-0,15);

$З_{осн}$ – основная заработная плата.

Получим:

$$З_{доп\ НР} = 0{,}12 * 15264 = 1832 \text{ руб.};$$

$$З_{доп\ И} = 0{,}12 * 98432 = 11812 \text{ руб.}$$

Заработная плата сотрудника без вычетов рассчитывается по формуле 9:

$$З_{ЗП} = З_{осн} + З_{доп}. \qquad (9)$$

Тогда:

$З_{ЗП} = 1832 + 15264 = 17096$ для НР,

$З_{ЗП} = 98432 + 11812 = 110244$ для И.

3.3.3. Отчисление во внебюджетные фонды (страховые отчисления)

В данной статье расходов отражаются обязательные отчисления по установленным законодательством Российской Федерации нормам органам государственного социального страхования (ФСС), пенсионного фонда (ПФ) и медицинского страхования (ФФОМС) от затрат на оплату труда работников.

Величина отчислений во внебюджетные фонды определяется исходя из следующей формулы 10:

$$З_{внеб} = k_{внеб} * (З_{осн} + З_{доп}), \quad (10)$$

где $k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.)

На 2014 г. в соответствии с Федеральным законом от 24.07.2009 №212-ФЗ установлен размер страховых взносов равный 30%. На основании пункта 1 ст.58 закона №212-ФЗ для учреждений, осуществляющих образовательную и научную деятельность в 2014 году водится пониженная ставка – 27,1%. Отчисления во внебюджетные фонды рекомендуется представлять в табличной форме (таблица 15).

Таблица 15 – расчет отчислений во внебюджетные фонды

Исполнитель	Основная ЗП, руб.	Дополнительная ЗП, руб.	Коэффициент отчислений во внебюджетные фонды	$З_{внеб}$, руб.
НР	15264	1832	0,271	4633
И	98432	11812	0,271	29876
Итого:				34509

Исходя из данных таблицы получили, что 34509 руб. уйдет во внебюджетные фонды.

3.3.4. Прочие расходы

Данная статья включает стоимость всех материалов, покупных изделий, полуфабрикатов и других материальных ценностей, расходуемых непосредственно в процессе выполнения работ. Цена материальных ресурсов определяется по соответствующим ценникам и приведена в таблице 16.

Таблица 16 –Затраты на оборудование и материалы

Наименование материалов	Цена за ед., руб.	Количество	Сумма, руб.
Бумага для принтера А4	170	1 шт.	170
Плата отладочная на базе микроконтроллера STM32	210	1 шт.	210
Внутрисхемный программатор/отладчик JTAG для мк STM8 и STM32	620	1 шт.	620
Кабель USB- Mini USB	120	1 шт.	120
Преобразователь интерфейсов FTDI232	270	1 шт.	270
Итого:			1390

Таким образом, расходы на материалы составляют 1390 руб.

Материальные затраты учитываются с учетом количества использованной электроэнергии. Для юридических лиц стоимость 1 кВт*ч составляет 5,8 рублей. При умеренном пользовании компьютер средней мощности затрачивает 1,8 кВт в день в среднем.

$$З_{мат} = 1{,}8\ кВт * 128\ дн. * 7\ ч * 5{,}8\ \frac{руб.}{кВт * ч} + З_{мат} = 9354\ руб. + 1390\ руб.$$

$$= 10744\ руб.$$

3.3.5. Формирование бюджета затрат научно-исследовательского проекта

Определение бюджета затрат на научно-исследовательский проект по каждому варианту исполнения приведен в таблице 17. Так как все виды исполнений включают одинаковое оборудование, отличаясь только программной реализацией, то данные в таблице будут одинаковыми.

Таблица 17 – Бюджет затрат по каждому исполнению НТИ

Наименование статьи	Сумма, руб. (Исп.1)	Сумма, руб. (Исп.2)	Сумма, руб. (Исп.3)
Материальные затраты НТИ и прочие расходы	10744	10744	10744
Затраты по основной заработной плате исполнителей	98432	98432	98432
Затраты по дополнительной заработной плате	11812	11812	11812
Отчисления во внебюджетные фонды	34509	34509	34509
Бюджет затрат НТИ	155497	155497	155497

3.4. Определение ресурсной, финансовой, бюджетной, социальной и экономической эффективности исследования

Научно-технический уровень характеризует, в какой мере выполнены работы и обеспечивается научно-технический прогресс в данной области. Для оценки научной ценности, технической значимости и эффективности, планируемых и выполняемых НИР, используется метод бальных оценок.

Таблица 18 – Сравнительная оценка характеристик вариантов исполнения проекта

Объект исследования / Критерии	Весовой коэффициент параметра	Исп.1	Исп.2	Исп.3
Способствует росту производительности труда	0,3	5	5	5
Удобство в эксплуатации (соответствует требованиям потребителей)	0,3	5	4	4
Помехоустойчивость	0,05	5	5	4
Энергосбережение	0,05	4	4	5
Надежность	0,15	5	5	4
Материалоемкость	0,15	5	5	4
Итого	1			

Рассчитаем интегральный показатель ресурсоэффективности по формуле 11:

$$I_{p\,i} = \sum a_i b_i \quad (11),$$

где, $I_{p\,i}$ – интегральный показатель ресурсоэффективности для i-го варианта исполнения разработки;

a_i – весовой коэффициент i-го варианта исполнения разработки;

b_i – бальная оценка i-го варианта исполнения разработки, устанавливается экспертным путем по выбранной шкале оценивания;

В итоге, исходя из данных таблицы

Таблица 18, получим:

$$I_{p\,\text{исп1}} = 4{,}95;$$

$$I_{p\,\text{исп2}} = 4{,}65;$$

$$I_{p\,\text{исп3}} = 4{,}35.$$

Интегральный показатель эффективности вариантов исполнения разработки ($I_{p\,i}$) определяется на основании интегрального показателя ресурсоэффективности и интегрального финансового показателя по формуле 12:

$$I_{\text{исп}\,i} = \frac{I_{p\,\text{исп}\,i}}{I_{\text{финр}}^{\text{исп}\,i}}, \quad (12)$$

Так как все исполнения имеют одинаковую стоимость, то $I_{\text{финр}}^{\text{исп}\,i}$=1 для каждого исполнения. Тогда интегральный показатель для исполнения 1 имеет наилучшую эффективность проекта и является наиболее целесообразным вариант из предложенных.

Сравнительная эффективность проекта определяется по формуле 13:

$$\text{Э}_{\text{ср}} = \frac{I_{\text{исп 1}}}{I_{\text{исп 2}}}, \quad (13)$$

Тогда получим:

$$\text{Э}_{\text{ср 1}} = \frac{4{,}95}{4{,}65} = 1{,}06$$

$$\text{Э}_{\text{ср 2}} = \frac{4{,}65}{4{,}65} = 1;$$

$$\text{Э}_{\text{ср 3}} = \frac{4{,}35}{4{,}65} = 0{,}94$$

Сравнительная я эффективность разработки представлена в таблице 19.

Таблица 19 – Сравнительная эффективность разработки

Показатель	Исп.1	Исп.2	Исп.3
Интегральный финансовый показатель разработки	1	1	1
Интегральный показатель ресурсоэффективности разработки	4,95	4,65	4,35
Интегральный показатель эффективности	4,95	4,65	4,35
Сравнительная эффективность вариантов исполнения	1,06	1	0,94

Исходя из полученных данных и проведенного анализа эффективности можно сделать вывод, что вариант исполнения 1 является наиболее эффективным с позиции финансовой и ресурсоэффективности.

4. Раздел «Социальная ответственность»

4.1. Аннотация

Представление понятия «Социальная ответственность» сформулировано в международном стандарте (МС) ICCSR-08260008000: 2011 «Социальная ответственность организации».

В соответствии с МС - Социальная ответственность - ответственность организации за воздействие ее решений и деятельности на общество и окружающую среду через прозрачное и этичное поведение, которое:

1. содействует устойчивому развитию, включая здоровье и благосостояние общества;

2. учитывает ожидания заинтересованных сторон;

3. соответствует применяемому законодательству и согласуется с международными нормами поведения (включая промышленную безопасность и условия труда, экологическую безопасность);

4. интегрировано в деятельность всей организации и применяется во всех ее взаимоотношениях (включая промышленную безопасность и условия труда, экологическую безопасность) [18].

4.2. Структура проекта

Основная идея устройства, созданного на базе микроконтроллера STM32, заключается в отделении органа управления от преобразователя частоты. В нашем случае основное управление и слежение за частотой вращения электродвигателем осуществляется через ПК. Список оборудования, используемого в проекте, представлен в таблице 20.

Таблица 20 – Оборудование, используемое в проекте

Наименование оборудования	Количество
Плата отладочная на базе микроконтроллера STM32	1 шт.
Внутрисхемный программатор/отладчик JTAG для мк STM8 и STM32	1 шт.

Наименование оборудования	Количество
Кабель USB- Mini USB	1 шт.
Преобразователь интерфейсов FTDI232	1 шт.

Созданное устройство обеспечивает взаимосвязь персонального компьютера и преобразователя частоты по проводному каналу связи (рисунок 27).

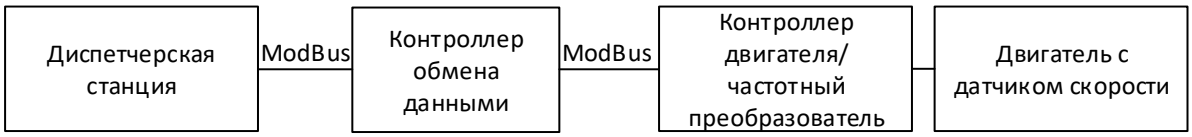


Рисунок 27 – Структурная схема проекта с проводным каналом связи

В дальнейшем планируется использовать беспроводной канал. В таком случае устройство, подключенное к ПЧ с одной стороны и к wi-fi модулю с другой, будет обмениваться данными с диспетчерской станцией по сети и выполнять функцию преобразования UART протокола в протокол ModBus. Это обуславливает необходимость микроконтроллера, как коммутирующего звена (рисунок 28).

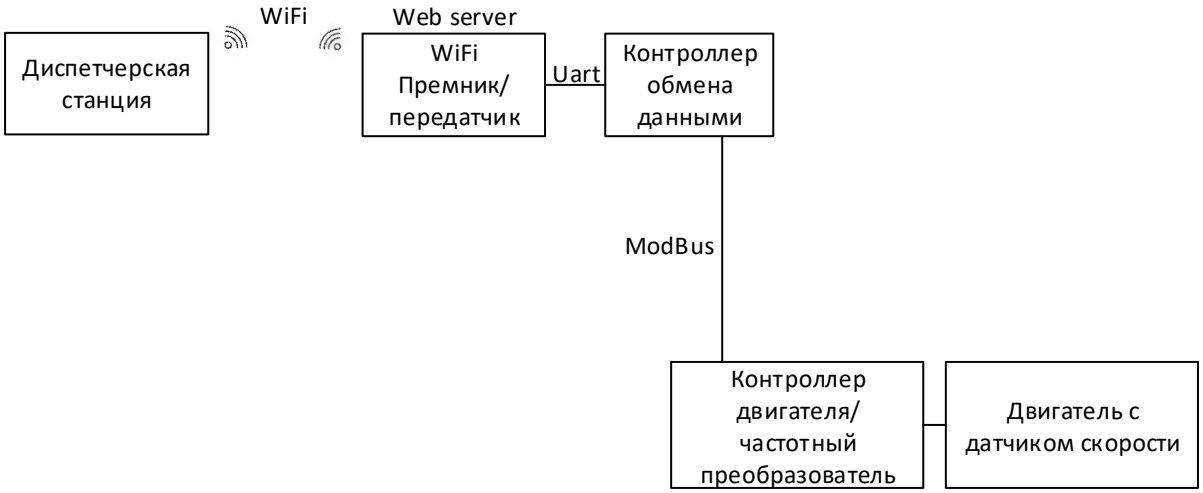


Рисунок 28 – Структурная схема проекта с беспроводным каналом связи

Это гарантирует безопасное расстояние диспетчерской станции от объекта управления, а также разрешает доступ к управлению объектом с любой точки мира, где есть выход в сеть.

4.3. Краткое описание протокола Modbus RTU

На текущем этапе отлажена работа устройства по проводному каналу. Для передачи данных используется протокол Modbus RTU. Modbus — протокол, работающий по принципу «клиент-сервер». Широко применяется в промышленности. Modbus может использоваться для передачи данных через последовательные линии связи RS-485, RS-422, RS-232, а также сети TCP/IP.

Контроллеры на шине Modbus взаимодействуют, используя masterslave модель, основанную на транзакциях, состоящих из запроса и ответа. Обычно в сети есть только одно ведущее, так называемое, «главное» (англ. master) устройство, и несколько ведомых — «подчинённых» (англ. slaves) устройств. Главное устройство (мастер) инициирует транзакции (передаёт запросы). Мастер может адресовать запрос индивидуально любому подчиненному или инициировать передачу широковещательного сообщения для всех подчиненных устройств. Подчинённое устройство, опознав свой адрес, отвечает на запрос, адресованный именно ему. При получении широковещательного запроса ответ подчинёнными устройствами не формируется.

Ведущим устройством в проекте является персональный компьютер, в качестве ведомого выступает преобразователь частоты. Так как к STM32 помимо частотного преобразователя может быть подключено несколько устройств, то, по правилам протокола, частотный преобразователь будет прерываться на обработку каждой команды, сверяя адрес устройства. В данной реализации протокола было принято решение переложить обязанность сравнения адресов на STM32.

Благодаря данной опции, конкретное ведомое устройство прерывается на обработку команд, адресованных конкретно ему. В нашем случае это позволяет повысить производительность частотного преобразователя и уменьшить риск его сбоя.

4.4. Помехоустойчивое кодирование в Modbus RTU

При передаче данных возможны различные помехи, помехой называют постороннее электрическое колебание, мешающее нормальному приему сигналов. Причиной и источниками помех могут являться различные факторы, и помехи могут быть классифицированы по различным признакам. В зависимости от места возникновения посторонние электрические колебания можно разделить на внешние и внутренние помехи. Внутренние помехи возникают в узлах аппаратуры и трактах систем связи. Внешние помехи обусловлены действием источников помех, внешних по отношению к системе связи и не связанных с ее функционированием.

В нашем случае большое влияние оказывают внутренние помехи. Для борьбы с ними, в программной среде для STM32 предусмотрена процедура проверки флагов устройства, которые указывают на различные неисправности. Но такая диагностика невозможна при удаленной работе устройства, поэтому, для проверки целостности поступающих данных, протокол Modbus использует CRC.

CRC — циклический избыточный код (англ. Cyclic redundancy check). Алгоритм нахождения контрольной суммы, предназначенный для проверки целостности данных. CRC является практическим приложением помехоустойчивого кодирования.

Существуют разные варианты нахождения CRC, важно, чтобы метод расчета CRC совпадал на всех устройствах.

В созданном проекте расчет CRC осуществляется табличным методом.

Далее представлен алгоритм проверки достоверности информации, реализованный на STM32 (рисунок 29).

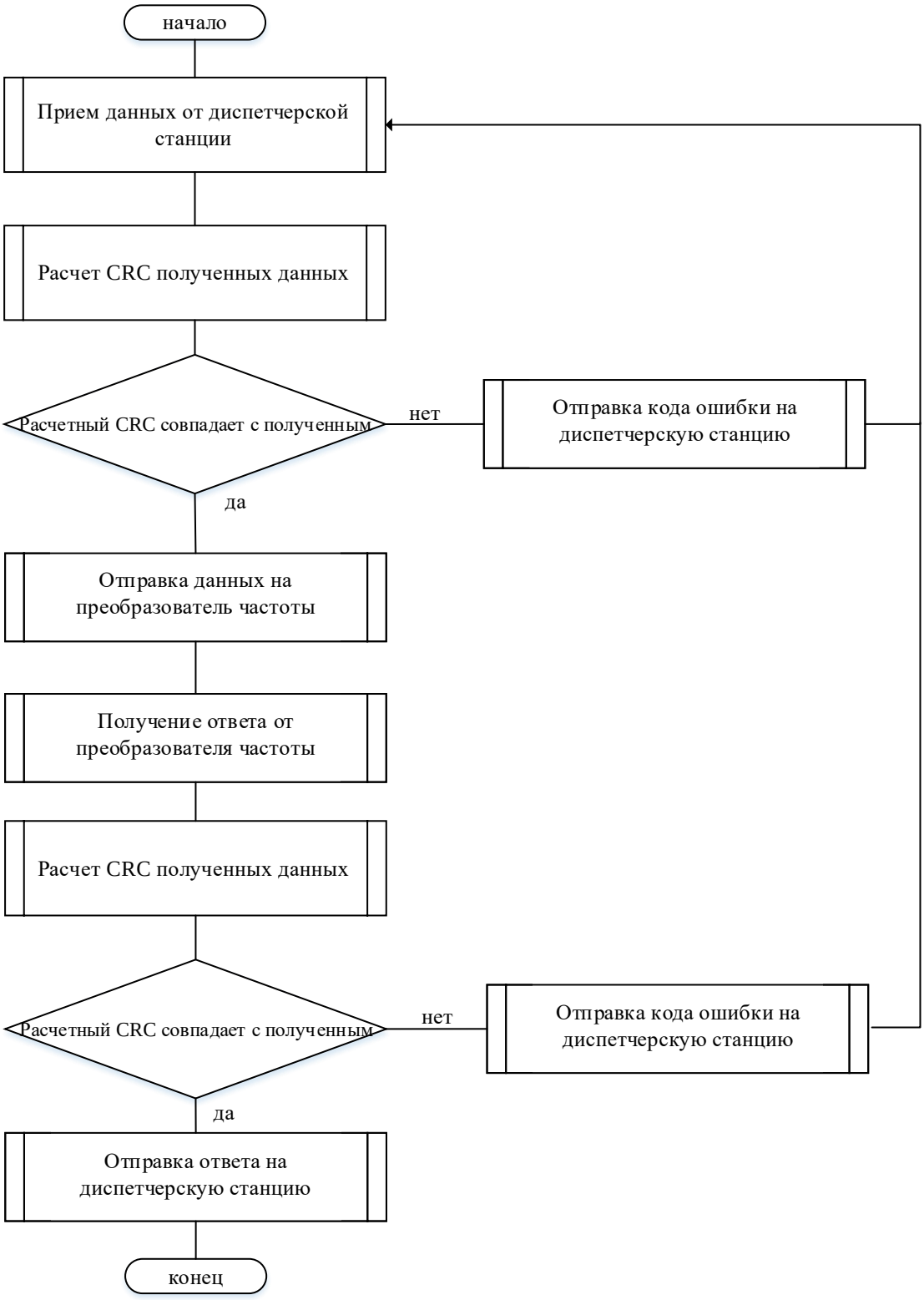


Рисунок 29 – Алгоритм проверки целостности полученных данных

4.5. Функция разгрузки информационного потока

Как упоминалось выше, протокол Modbus предполагает отправку команд с ведущего устройства всем ведомым устройствам В структуре команды веду-

щего устройства есть поле адреса ведомого устройства, которому предназначается команда (таблица 21).

Таблица 21 – Формат команды запроса на чтение данных

ID ведомого устройства	Идентификатор команды	Данные				CRC-16	
0x01	0x03	0x00	0x02	0x00	0x02	0xC5	0xCD

Каждое ведомое устройство обязано сверить адрес полученной команды со своим и, в случае успеха, выполнить команду. Тем самым, при отправке любой команды с ведущего устройства, все ведомые устройства выполняют проверку этой команды на соответствие адреса.

Для уменьшения информационного потока на ведомые устройства, в STM32 была создана функция, осуществляющая селекцию ведомых устройств в зависимости от адреса, указанного в команде ведущего устройства. Иными словами, созданная функция разгружает работу ведомых устройств и гарантирует, что на каждое ведомое устройство придет команда, предназначенная именно ему.

Сопоставление адресов в командах и каналов, подключенных ведомых устройств производится в теле программы микроконтроллера с помощью оператора ветвления CASE. Структурная схема алгоритма представлена на рисунке 30.

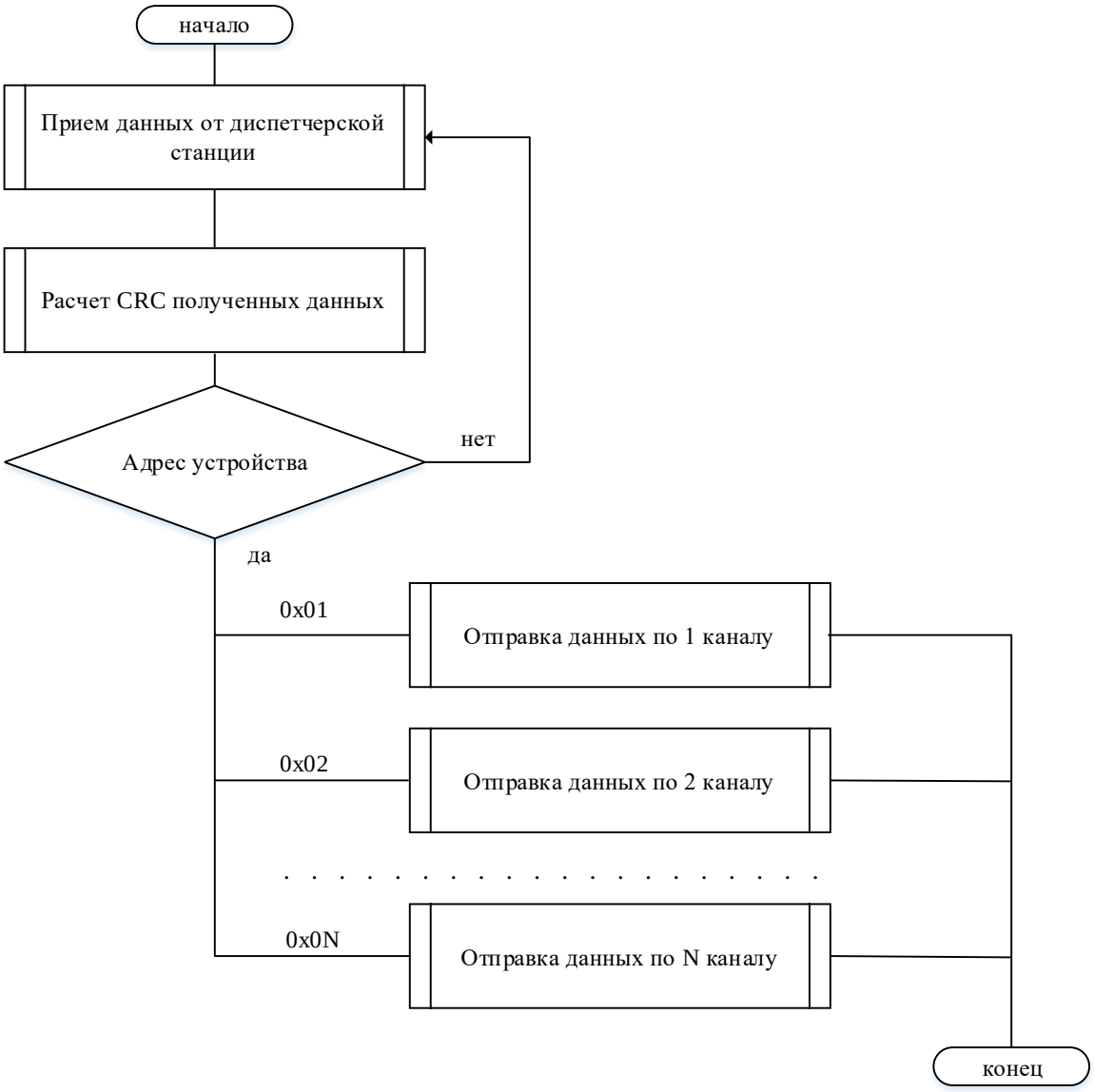


Рисунок 30 – Алгоритм разгрузки информационного потока на ведомые устройства

4.6. Резервирование и самодиагностика

В зависимости от сферы применения созданной системы управления возможно повышение ее надежности с помощью резервирования. Составной частью систем с резервированием является подсистема автоматического контроля работоспособности и диагностики неисправностей.

В случае, когда система некритична к времени простоя величиной в несколько минут достаточно использовать холодное резервирование. Ненагруженный резерв («холодный» резерв) — резервный элемент, находящийся в ненагруженном режиме до начала его использования вместо основного элемента.

Ненагруженный резерв позволяет получить системы с самой высокой надёжностью, но с низким коэффициентом готовности.

В системах, технологический процесс которых не позволяет простоя в отслеживании и регулировании параметров стоит использовать нагруженный резерв («горячий» резерв) — резервный элемент, который находится в таком же режиме, как и основой. В системах автоматизации с «горячим» резервом переход на резерв может занимать время от нескольких миллисекунд до единиц секунд. Для обеспечения «горячей» замены необходимо предусмотреть следующее:

- защиту от статического электричества, которое может возникать на теле оператора, выполняющего замену устройства;
- необходимую последовательность подачи напряжений питания и внешних сигналов (для этого используют, например, разъёмы с контактами разной длины и секвенсоры внутри устройства);
- защиту системы от броска тока, вызванного зарядом ёмкостей подключаемого устройства, например, с помощью токоограничительных резисторов или отдельного источника питания;
- защиту устройства от перенапряжения, короткого замыкания, превышения напряжения питания, ошибочного подключения.

Помимо резервирования микроконтроллеров, участвующих в работе, особое внимание стоит уделить элементам памяти, хранящим историю протекания процесса и архивированию этих данных.

Существуют также методы увеличения надежности, на основе самодиагностики. Например, во FLASH памяти можно иметь две прошивки МК, каждая из которых будет проверяться перед стартом на контрольную сумму (некоторое значение, рассчитанное по набору данных путём применения определённого алгоритма и используемое для проверки целостности данных при их передаче или хранении) и запускаться только в случае ее соответствия. В итоге, полученная система является более надежной и, даже в случае единичной ошибки во FLASH памяти, она сможет запуститься с резервной копии.

Заключение

В данной выпускной квалификационной работе были выполнены задачи по изучению системы управления частотой электродвигателя. Разработана система управления, включающая в себя сопрягающее устройство на основе микроконтроллера STM32. Устройство работает в двунаправленном режиме, обрабатывая и принимая данные с диспетчерской станции и преобразователя частоты. Для программирования устройства было изучено программное обеспечение: графический генератор кода STM32CubeMX и среда разработки Keil uVision. При программировании в среде Keil uVision основное внимание было уделено изучению функциональных возможностей библиотеки, на которой базируется сгенерированный в STM32CubeMX код.

При реализации протокола Modbus RTU, отдельной задачей стояла задача реализации функции приема данных, поскольку библиотечная функция принимает только фиксированное количество байт, а в разрабатываемой системе принимались пакеты различной длины. Для этого был организован прием данных на уровне регистров. Это позволило включить операторы ветвления после каждого ключевого байта (байт кода функции), благодаря этому был реализован гибкий алгоритм, принимающий пакеты различной длины.

Для реализации сопрягающего устройства использовались разработанные ранее программы протоколов UART и Modbus. При тестировании устройства возникали ошибки, связанные с физическим нарушением данных в каналах связи. Обнаружить данные проблемы удалось с помощью отладчика Keil uVision, в котором можно просматривать регистры флагов микроконтроллера, которые и указали на причину проблемы.

Введение функции разгрузки в устройства сопряжения изменило топологию обмена данными по протоколу Modbus с «общей шины» на иерархическую, однако, это не создало неудобств при приеме/передачи данных между ведущим и ведомыми устройствами, поскольку ресурсы STM32 позволяют обрабатывать большой поток информации. В дальнейшем планируется введение функции

преобразования протоколов, получая известный протокол на входе, устройство будет преобразовывать данные в заданный протокол на выходе. Также планируется реализовать алгоритм обмена данными устройства с WI-FI модулем, тогда отслеживание параметров и управление электродвигателем будет возможно удаленно от места нахождения двигателя. Это позволит изолировать диспетчерскую станцию от технологического процесса.

Список литературы

1. Две стороны повсеместного применения микроконтроллеров [Электронный ресурс]/Хабр: https://habr.com/post/137987/, свободный. Яз. Рус. Дата обращения: 27.05.2018.

2. Управление двигателями постоянного тока [Электронный ресурс]/ РадиоЛоцман: https://www.rlocman.ru/shem/schematics.html?di=157614, свободный. Яз. Рус. Дата обращения: 27.05.2018.

3. Способы управления электродвигателями [Электронный ресурс]/ Инженерные решения: http://engineering-solutions.ru/motorcontrol/techniques/, свободный. Яз. Рус. Дата обращения: 27.05.2018.

4. Modbus, протокол: описание, сфера применения, достоинства и недостатки [Электронный ресурс]/FB: http://fb.ru/article/349993/modbus-protokol-opisanie-sfera-primeneniya-dostoinstva-i-nedostatki, свободный. Яз. Рус. дата обращения: 27.05.2018.

5. STM32 и Arduino: сравнение характеристик, плюсы и минусы [Электронный ресурс]/Arduino+: https://arduinoplus.ru/stm32-i-arduino-sravnenie/, свободный. Яз. Рус. дата обращения: 27.05.2018.

6. UART протокол [Электронный ресурс]/Хабр: https://habr.com/post/109395/, свободный. Яз. Рус. Дата обращения: 27.05.2018.

7. Прямой доступ к памяти [Электронный ресурс]/Википедия: https://ru.wikipedia.org/wiki/Прямой_доступ_к_памяти, свободный. Яз. Рус. Дата обращения: 27.05.2018.

8. STM32 и Arduino: сравнение характеристик, плюсы и минусы [Электронный ресурс]/Arduino+: https://arduinoplus.ru/stm32-i-arduino-sravnenie/, свободный. Яз. Рус. Дата обращения: 27.05.2018.

9. Программные средства разработки ПО для STM32 [Электронный ресурс]/Компэл: https://www.compel.ru/lib/ne/2017/1/8-prostyie-besplatnyie-programmnyie-sredstva-razrabotki-po-dlya-stm32, свободный. Яз. Рус. Дата обращения: 27.05.2018.

10. Управление двигателями постоянного тока [Электронный ресурс] URL: http://www.ti.com/lit/an/tida012/tida012.pdf, свободный. Яз. Рус. Дата обращения: 27.05.2018.

11. STM32CubeMX - With Example Work-flow [Электронный ресурс]/FreeRTOS: https://www.freertos.org/FreeRTOSPlus/BSP_Solutions/ST/STM32CubeMX.html, свободный. Яз. Рус. Дата обращения: 27.05.2018.

12. STM32Cube Создание проекта [Электронный ресурс]/MicroTechnics: http://microtechnics.ru/stm32cube-sozdanie-proekta/, свободный. Яз. Рус. Дата обращения: 27.05.2018.

13. Визуализация возможностей: графический генератор кода STM32CubeMX [Электронный ресурс]/Компэл: https://www.compel.ru/lib/ne/2014/11/4-vizualizatsiya-vozmozhnostey-graficheskiy-generator-koda-stm32cubemx, свободный. Яз. Рус. Дата обращения: 27.05.2018.

14. Keil uVision [Электронный ресурс]/cxem.net: http://cxem.net/software/keil.php, свободный. Яз. Рус. Дата обращения: 27.05.2018.

15. UART и USART [Электронный ресурс]/Hamper's site: http://www.rotr.info/electronics/mcu/arm_usart.htm, свободный. Яз. Рус. Дата обращения: 27.05.2018.

16. CRC16 - Описание алгоритма и пример расчёта [Электронный ресурс]/PIClist: http://piclist.ru/S-CRC16-RUS/CRC16.html, свободный. Яз. Рус. Дата обращения: 27.05.2018.

17. [Электронный ресурс]/EasySTM32: http://easystm32.ru/for-beginners/25-interrupts-handling-in-stm32, свободный. Яз. Рус. Дата обращения: 27.05.2018.

18. Романенко С.В., Анищенко Ю.В. Социальная ответственность: методическое указание / Томский политехнический университет. - Томск: Изд-во Томского политехнического университета, 2016. -21 с.

Приложение А.

(основное)
Листинг функции расчета CRC-16

```
uint16_t CRC16(uint8_t *p, uint16_t len) // инициализация функции расчета CRC
{
const uint8_t auchCRCLo[256]=  // инициализация таблиц
{
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5,
0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9,
0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F,
0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13,
0xD3, 0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6,
0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA,
0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA,
0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6,
0x26, 0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3,
0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF,
0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79,
0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75,
0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90,
0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C,
0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C,
0x8C, 0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81,
0x80,0x40
};
const uint8_t auchCRCHi[256]=
{
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80,
0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80,
0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80,
0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40   };
```

```
uint8_t crc_hi, crc_lo, n;
if (len>256U) return (0);
n = (uint8_t)len;
crc_hi = 0xFF;                  // зануляем старший байт CRC
crc_lo = 0xFF;                  // зануляем младший байт CRC
do
{
uint8_t i = crc_hi ^ *p++;
crc_hi = crc_lo ^ (uint8_t)auchCRCHi[i];    // расчет  CRC
crc_lo =          (uint8_t)auchCRCLo[i];
}
while (--n);
return ((crc_hi << 8) | crc_lo);
}
```

Приложение Б.

(основное)

Листинг приема/отправки информации по UART протоколу

```
int main(void)
{
 /* USER CODE BEGIN 1 */
uint8_t str[2]; // объявление переменной str
 /* USER CODE BEGIN WHILE */
 while (1)
 {
   HAL_Delay(100); // задержка 100мс
      if(huart1.RxXferCount==0) // счетчик USART, отвечающий за количество принятых байт
        {
                 HAL_UART_Receive(&huart1,(uint8_t*) str,1,100); // прием байта
                HAL_UART_Transmit(&huart1,(uint8_t*) str,1,100); // отправка байта
                str[0]=0; // обнуление переменной
        }
 /* USER CODE END WHILE */
 /* USER CODE BEGIN 3 */
 }
 /* USER CODE END 3 */
}
```

Приложение В.

(основное)
Листинг обмена данными по протоколу Modbus

```
void USART2_IRQHandler(void)
{
 /* USER CODE BEGIN USART2_IRQn 0 */

        if(__HAL_UART_GET_FLAG(&huart2, UART_FLAG_RXNE)) // флаг прерывания по приему
                {
                        USART2->CR1&=~USART_CR1_TE;        // Выключаем Tx
                        USART2->SR&=~USART_SR_RXNE; // Сбрасываем флаг прерывания
                        rxar[b]=USART2->DR;    // Присваиваем байт информации элементу массива
                        b++;                                // Счетчик принятых байтов

                        // Обработка запроса на чтение и отправка посылки
                        if( rxar[1]==0x03)
                                {
                                        if (b==8)
                                        {
                                            // Проверка на целостность данных
                                            if((rxar[6]==(CRC16(rxar,6)>>8))&&(rxar[7]==(CRC16(rxar,6)&0xFF)))
                                                {
                                                        txar_read[0]=rxar[0];               // создание посылки
                                                        txar_read[1]=rxar[1];
                                                        txar_read[2]=0x02;
                                                        txar_read[3]=0x00;
                                                        txar_read[4]=data_for_read[rxar[3]];
                                                        txar_read[5]=CRC16(txar_read,5)>>8;
                                                        txar_read[6]=CRC16(txar_read,5)&0xFF;

                                                        b=0;                // обнуление счетчика

                                                        USART2->CR1|=USART_CR1_TE; // Включаем Tx

                                                        // Отправка данных
                                                        HAL_UART_Transmit_IT(&huart2,(uint8_t*) txar_read,7);


                                                }
                                            else
                                                {
                                                        for (int i=0; i<8; i++) rxar[i]=0;    // обнуление массива
                                                        b=0;                                // обнуление счетчика
                                                }
                                        }

                                }
```

```
        // Обработка запроса на запись и отправка посылки
        if (rxar[1]==0x10)
                {
                   if (b==13)c
                        {
                         // Проверка на целостность данных
                          if ((rxar[11]==(CRC16(rxar,11)>>8))&&(rxar[12]==(CRC16(rxar,11)&0xFF)))
                                {
                                        txar_write[0]=rxar[0];   // создание посылки
                                        txar_write[1]=rxar[1];
                                        txar_write[2]=rxar[2];
                                        txar_write[3]=rxar[3];
                                        txar_write[4]=rxar[4];
                                        txar_write[5]=rxar[5];
                                        txar_write[6]=CRC16(txar_write,6)>>8;
                                        txar_write[7]=CRC16(txar_write,6)&0xFF;

                                        rec_1[rxar[3]]=rxar[7];  // Записываем полученные данные
                                        rec_2[rxar[3]]=rxar[8];
                                        rec_3[rxar[3]]=rxar[9];
                                        rec_4[rxar[3]]=rxar[10];

                                        b=0;                        // обнуление счетчика

                                        USART2->CR1|=USART_CR1_TE;                    // Включаем Tx

                                        // Отправка данных
                                        HAL_UART_Transmit_IT(&huart2,(uint8_t*) txar_write,8);
                                }
                          else
                                {
                                        for (int i=0; i<13; i++) rxar[i]=0; // обнуление массива
                                        b=0;                                // обнуление счетчика
                                }

                          }
                        }
                }
 HAL_UART_IRQHandler(&huart2);                // встроенный обработчик прерывания

}
```

Приложение Г.

(основное)
Листинг работы устройства сопряжения

```
void USART2_IRQHandler(void)
{
  if(__HAL_UART_GET_FLAG(&huart2, UART_FLAG_RXNE)) // флаг прерывания по приему
        {
          USART1->CR1&=~USART_CR1_TE;     // Выключаем Tx
          USART2->SR&=~USART_SR_RXNE; // Сбрасываем флаг прерывания
          rxar[b]=USART2->DR;           // Присваиваем байт информации элементу массива
          b++;                          // Счетчик принятых байтов
          // Обработка запроса на чтение и отправка посылки
          if( rxar[1]==0x03)
                {
                  if (b==8)
                        {
                        // Проверка на целостность данных
                        if ((rxar[6]==(CRC16(rxar,6)>>8))&&(rxar[7]==(CRC16(rxar,6)&0xFF)))
                        {
                             b=0;          // обнуление счетчика
                        // функция включения нужного канала UART в зависимости от адреса устройства
                            switch (rxar[0])
                                {
                                   case 0x01:
                                    USART1->CR1|=USART_CR1_TE;     // Включаем Tx
                                    // Отправка данных
                                    HAL_UART_Transmit_IT(&huart1,(uint8_t*)rxar,8);
                                   break;
                                }

                                }
                        else
                                {
                                   for (int i=0; i<8; i++) rxar[i]=0;                    // обнуление массива
                                   b=0;                                                  // обнуление счетчика
                                }
                                }
                }
                // Обработка запроса на запись и отправка посылки
                   if (rxar[1]==0x10)
                        {
                          if (b==13)
                                {
                                 // Проверка на целостность данных
                                 if ((rxar[11]==(CRC16(rxar,11)>>8))&&(rxar[12]==(CRC16(rxar,11)&0xFF)))
                                        {
                                                b=0;              // обнуление счетчика
```

```
    // функция включения нужного канала UART в зависимости от адреса устройства
                                        switch (rxar[0])
                                          {
                                              case 0x01:
                                                 USART1->CR1|=USART_CR1_TE;          // Включаем Tx
                                                 // Отправка данных
                                                 HAL_UART_Transmit_IT(&huart1,(uint8_t*) rxar,13);
                                                break;
                                           }
                                        }
                                  else
                                        {
                                          for (int i=0; i<8; i++) rxar[i]=0;                  // обнуление массива
                                           b=0;                                        // обнуление счетчика
                                        }

                                  }
                        }
                }
 HAL_UART_IRQHandler(&huart2);                // встроенный обработчик прерывания
}

void USART3_IRQHandler(void)
{
        if(__HAL_UART_GET_FLAG(&huart3, UART_FLAG_RXNE)) // флаг прерывания по приему
                {
                        USART2->CR1&=~USART_CR1_TE; // Выключаем Tx
                        USART3->SR&=~USART_SR_RXNE; // Сбрасываем флаг прерывания
                        rxar3[b1]=USART3->DR;          // Присваиваем байт информации элементу массива
                        b1++;                          // Счетчик принятых байтов

                        if( rxar3[1]==0x03)
                                {
                                        if (b1==7)
                                        {
                                        // Проверка на целостность данных
                                        if((rxar3[5]==(CRC16(rxar3,5)>>8))&&(rxar3[6]==(CRC16(rxar3,5)&0xFF)))
                                                {
                                                   b1=0;                    // обнуление счетчика
                                                   USART2->CR1|=USART_CR1_TE;       // Включаем Tx
                                                   // Отправка данных
                                                   HAL_UART_Transmit_IT(&huart2,(uint8_t*) rxar3,7);
                                        }
                                         else
                                                {
                                                   for (int i=0; i<8; i++) rxar[i]=0;          // обнуление массива
                                                   b1=0;                                       // обнуление счетчика
                                                        }
                                        }
                                }
```

```
                    if (rxar3[1]==0x10)
                       {
                          if (b1==8)
                             {
                             // Проверка на целостность данных
                             if ((rxar3[6]==(CRC16(rxar3,6)>>8))&&(rxar3[7]==(CRC16(rxar3,6)&0xFF)))
                                    {
                                      b1=0;                       // обнуление счетчика
                                      USART2->CR1|=USART_CR1_TE;        // Включаем Tx
                                      // Отправка данных
                                      HAL_UART_Transmit_IT(&huart2,(uint8_t*) rxar3,8);
                                    }
                             else
                                    {
                                      for (int i=0; i<8; i++) rxar[i]=0;          // обнуление массива
                                      b1=0;                                       // обнуление счетчика
                                    }

                              }
                       }
              }

  HAL_UART_IRQHandler(&huart3); // встроенный обработчик прерывания

}
```

Приложение Д.

(основное)
Карта сегментирования рынка

		Управляющие устройства			
		Промышленные контроллеры Schneider Electric	Промышленные контроллеры Siemens	Микроконтроллеры STM32	Микроконтроллеры на базе Arduino
Потребители	Крупные предприятия				
	Средние предприятия				
	Мелкие предприятия				

Приложение Е.

(основное)

Временные показатели научного исследования

№ работ	Название работы	Исполнители	$T_к$, дни	Продолжительность выполнения работ														
				Январь			Февраль			Март			Апрель			Май		
				1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	Составление и утверждение технического задания	2	4															
2	Календарное планирование работ	2	1															
3	Изучение возможного ПО для программирования STM32	1	3															
4	Выбор наиболее подходящего ПО и его углубленное изучение	1	6															
5	Отправка и прием информации на STM32, используя интерфейс передачи данных USART	1	18															
6	Тестирование работы при разных скоростях передачи	1	13															
7	Изучение правил приема передачи данных	1	7															
8	Изучение табличного способа расчета CRC	1	3															
9	Изучение функции чтения и функции записи протокола Modbus RTU	1	15															
10	Программирование протокола и его отладка	1	20															
11	Создание функции выбора устройства на STM32	1	13															
12	Отладка работы устройства, запуск частотного преобразователя	1	18															
13	Создание отчета по проектной работе	2	7															

— Студент — Руководитель